Trabajo3

Carlos Morales Aguilera
David Criado Ramón

26 de mayo de 2017

CLASIFICACIÓN - EMAIL SPAM

Procedemos a leer los datos del problema de clasificación **Email Spam**, el cual consiste en determinar dada una serie de características sobre un mensaje, si este es correo basura o no.

Procedemos inicialmente a la lectura de datos excluyendo de las muestras aquellos que tengan alguna característica con un valor perdido.

```
# Establecemos la semilla aleatoria
set.seed(11)

# Cambiamos de directorio
setwd("./datos")

# Lectura del fichero
spam = read.table("spam.data",
                  quote="\"", comment.char="", stringsAsFactors=FALSE)

# Salimos del directorio de los datos
setwd("..")

# Eliminación de valores perdidos
spam = na.omit(spam)
```

A continuación, vamos a crear una función en R que nos permita particionar los datos en conjuntos de entrenamiento, validación y test. Para ello indicamos el porcentaje que queremos de cada subconjunto y los datos, obteniendo así sus correspondientes subconjuntos de índices.

```
# Función que particiona los datos dados unos porcentajes
# y devuelve subconjuntos de índices
particionar = function(porcentajeTraining=2/3, porcentajeValidacion=0, data) {
    # Obtenemos un subconjunto de índices aleatorios que correspondera al training
    training = sample(1:nrow(data), porcentajeTraining*nrow(data))

    # Cogemos aquellos índices que no hayan sido utilizados para el subconjunto
    # de test
    test = (1:nrow(data))[-training]

    # Cogemos de training un porcentaje para el conjunto de validación
    validacion = sample(training,porcentajeValidacion*length(training))
    if (length(validacion) != 0) training = training[-validacion]

    # Devolvemos una lista con los tres subconjuntos
    list(training = training, test = test, validacion = validacion)
}
```

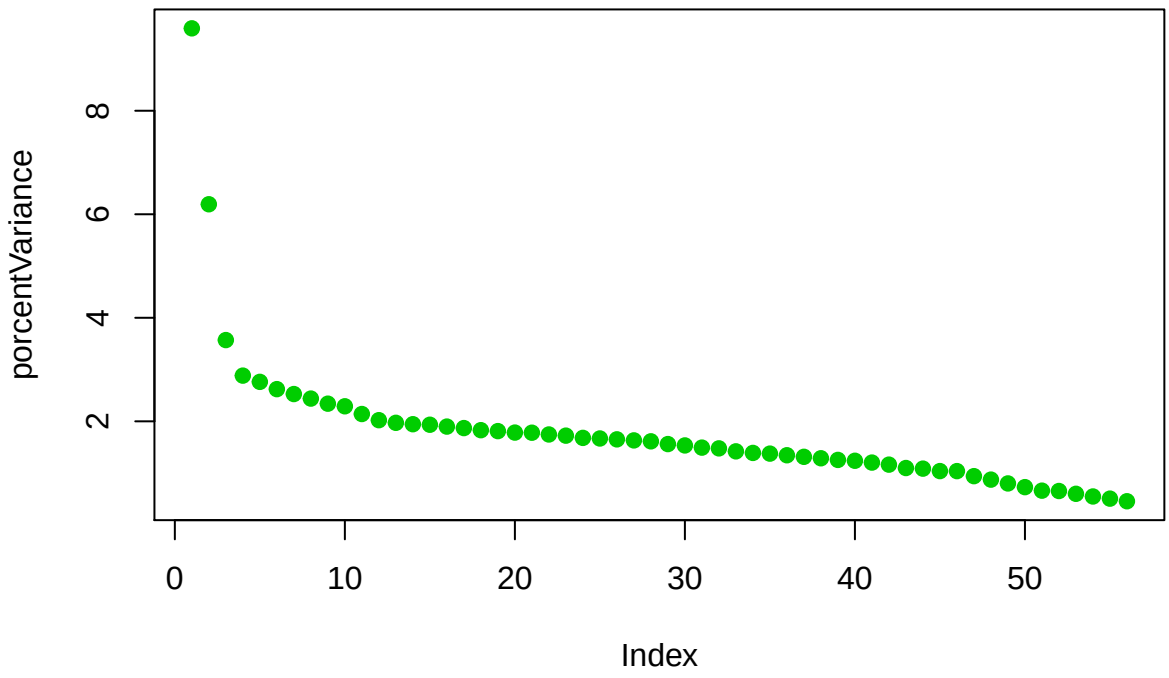
```
# Realizamos la partición de datos
i = particionar(data = spam)
```

Procedemos a preprocesar los datos:

- Eliminamos las variables con un grado de correlación superior al 75%.
- Con los datos ya escalados y centrados, utilizamos el filtro *PCA* para reducir la dimensionalidad, eliminando atributos irrelevantes o redundantes. Como observamos en la gráfica posterior que no hay una gran diferencia entre la varianza de los distintos atributos, y nos interesa un modelo lo más simple posible, nos quedamos con el número de atributos necesario para justificar el 80% de la varianza.
- Utilizamos *BoxCox* para transformar los atributos asimétricos reduciendo así la sensibilidad a la distancia entre ellos.

```
# 1. ELIMINAMOS LAS VARIABLES CON UN ALTO GRADO DE CORRELACIÓN.
# Obtenemos la matriz de correlaciones de los datos de training
descrCor = cor(spam[i$training,])
# Cogemos aquellos atributos cuyo grado de correlación es mayor al 75%
highCor = findCorrelation(descrCor, cutoff = .75)
# Quitamos dichos atributos
if (length(highCor) != 0) spam = spam[,-highCor]

# 2. UTILIZAMOS EL FILTRO PCA PARA REDUCIR LA DIMENSIONALIDAD
# Con los datos escalados y centrados, aplicamos el algoritmo PCA
pcaObject = prcomp(spam,center = TRUE, scale. = TRUE)
# Calculamos el porcentaje de la varianza y lo mostramos en una gráfica
porcentVariance = pcaObject$sd^2/sum(pcaObject$sd^2)*100
plot(porcentVariance, type="p", pch = 19, col = 19)
```




```
# 3. REALIZAMOS EL PREPROCESAMIENTO CON LAS TRANSFORMACIONES
# PREVIAMENTE MENCIONADAS Y EL BOXCOX
# Calculamos la transformación del preprocesamiento basándonos en training
preProc = preProcess(x = spam[i$training,1:(ncol(spam)-1)],
                     method = c("BoxCox", "center", "scale", "pca"),
                     outcome = spam[i$training, ncol(spam)],
                     thresh = 0.8
                     )

# Aplicamos la transformación a los subconjuntos.
training = predict(preProc, spam[i$training,1:(ncol(spam)-1)])
test = predict(preProc, spam[i$test,1:(ncol(spam)-1)])
```

Llegados a este punto, vamos a utilizar la función *regsubsets*, la cual se encarga de ir buscando las combinaciones de variables (ya preprocesadas) más significativas de las disponibles en el conjunto de datos.

A continuación aplicamos regresión logística con y sin regularización, obteniendo el error de cada una aplicando *bootstrapping* para obtener 25 muestras de entrenamiento por modelo y seleccionando aquel que tenga mayor precisión (*accuracy*).

```
# Realizamos la selección de modelos
formulas = regsubsets(x=training, y = spam[i$training, ncol(spam)])
# Mostramos la tabla de modelos a probar
summary(formulas)$outmat
```

```
##          PC1 PC2 PC3 PC4 PC5 PC6 PC7 PC8 PC9 PC10 PC11 PC12 PC13 PC14 PC15
## 1  ( 1 ) " " "*" " " " " " " " " " " " " " " " "  " "  " "  " "  " "  " "
## 2  ( 1 ) "*" "*" " " " " " " " " " " " " " " " "  " "  " "  " "  " "  " "
```

```
## 3  ( 1 ) "*" "*" "*" " " " " " " " " " " " " " " " "  " "  " "  " "  " "  " "
## 4  ( 1 ) "*" "*" "*" " " " " " " " " " " " " " " " "  "*"  " "  " "  " "  " "
## 5  ( 1 ) "*" "*" "*" "*" " " " " " " " " " " " " " "  "*"  " "  " "  " "  " "
## 6  ( 1 ) "*" "*" "*" "*" " " " " " " "*" " " " "  "*"  " "  " "  " "  " "
## 7  ( 1 ) "*" "*" "*" "*" " " "*" " " "*" " " " "  "*"  " "  " "  " "  " "
## 8  ( 1 ) "*" "*" "*" "*" " " "*" " " "*" " " " "  "*"  " "  " "  " "  " "
##          PC16 PC17 PC18 PC19 PC20 PC21 PC22 PC23 PC24 PC25 PC26 PC27 PC28
## 1  ( 1 ) " "  " "  " "  " "  " "  " "  " "  " "  " "  " "  " "  " "  " "
## 2  ( 1 ) " "  " "  " "  " "  " "  " "  " "  " "  " "  " "  " "  " "  " "
## 3  ( 1 ) " "  " "  " "  " "  " "  " "  " "  " "  " "  " "  " "  " "  " "
## 4  ( 1 ) " "  " "  " "  " "  " "  " "  " "  " "  " "  " "  " "  " "  " "
## 5  ( 1 ) " "  " "  " "  " "  " "  " "  " "  " "  " "  " "  " "  " "  " "
## 6  ( 1 ) " "  " "  " "  " "  " "  " "  " "  " "  " "  " "  " "  " "  " "
## 7  ( 1 ) " "  " "  " "  " "  " "  " "  " "  " "  " "  " "  " "  " "  " "
## 8  ( 1 ) " "  " "  " "  "*"  " "  " "  " "  " "  " "  " "  " "  " "  " "
##          PC29 PC30 PC31 PC32 PC33 PC34
## 1  ( 1 ) " "  " "  " "  " "  " "  " "
## 2  ( 1 ) " "  " "  " "  " "  " "  " "
## 3  ( 1 ) " "  " "  " "  " "  " "  " "
## 4  ( 1 ) " "  " "  " "  " "  " "  " "
## 5  ( 1 ) " "  " "  " "  " "  " "  " "
## 6  ( 1 ) " "  " "  " "  " "  " "  " "
## 7  ( 1 ) " "  " "  " "  " "  " "  " "
## 8  ( 1 ) " "  " "  " "  " "  " "  " "
```

```r
# Pasamos la matriz preprocesada a un dataframe
training_df = data.frame(training)
training_df["y"] = as.factor(spam[i$training, ncol(spam)])

# Creamos la matriz de precisiones de modelos
modelos = matrix(nrow = 5, ncol = 2)

# Función que da la precisión de un modelo regularizado para su mejor lambda y alpha
obtenerAccuracy = function(modelo) {
  modelo$results$Accuracy[modelo$results$lambda == modelo$bestTune$lambda &
                          modelo$results$alpha  == modelo$bestTune$alpha]
}

# Ajustamos los modelos
modelo1 = train(y ~ PC1 + PC2 + PC3 + PC11, data = training_df,
                     method ="glm", family ="binomial")
modelo1reg = train(y ~ PC1 + PC2 + PC3 + PC11,
                   data = training_df, method ="glmnet", family ="binomial")
modelo2 = train(y ~ PC1 + PC2 + PC3 + PC4 + PC11, data = training_df,
                     method ="glm", family ="binomial")
modelo2reg = train(y ~ PC1 + PC2 + PC3 + PC4 + PC11,
                     data = training_df, method ="glmnet", family ="binomial")
modelo3 = train(y ~ PC1 + PC2 + PC3 + PC4 + PC8 + PC11, data = training_df,
                     method ="glm", family ="binomial")
modelo3reg = train(y ~ PC1 + PC2 + PC3 + PC4 + PC8 + PC11,
                   data = training_df, method ="glmnet", family ="binomial")
modelo4 = train(y ~ PC1 + PC2 + PC3 + PC4 + PC6 + PC8 + PC11,
    data = training_df, method ="glm", family ="binomial")
```

```
modelo4reg = train(y ~ PC1 + PC2 + PC3 + PC4 + PC6 + PC8 +
          PC11, data = training_df, method ="glmnet", family ="binomial")

modelo5 = train(y ~ PC1 + PC2 + PC3 + PC4 + PC6 + PC8 + PC11 + PC19,
    data = training_df, method ="glm", family ="binomial")

modelo5reg = train(y ~ PC1 + PC2 + PC3 + PC4 + PC6 + PC8 + PC11
      + PC19, data = training_df, method ="glmnet", family ="binomial")


# Rellenamos la matriz con las precisiones de los modelos
modelos[1,1] = modelo1$results$Accuracy
modelos[1,2] = obtenerAccuracy(modelo1reg)
modelos[2,1] = modelo2$results$Accuracy
modelos[2,2] = obtenerAccuracy(modelo2reg)
modelos[3,1] = modelo3$results$Accuracy
modelos[3,2] = obtenerAccuracy(modelo3reg)
modelos[4,1] = modelo4$results$Accuracy
modelos[4,2] = obtenerAccuracy(modelo4reg)
modelos[5,1] = modelo5$results$Accuracy
modelos[5,2] = obtenerAccuracy(modelo5reg)

rownames(modelos) = c("(1) PC1 + PC2 + PC3 + PC11",
                      "(2) PC1 + PC2 + PC3 + PC4 + PC11",
                      "(3) PC1 + PC2 + PC3 + PC4 + PC8 + PC11",
                      "(4) PC1 + PC2 + PC3 + PC4 + PC6 + PC8 + PC11",
                      "(5) PC1 + PC2 + PC3 + PC4 + PC6 + PC8 + PC11 + PC19")

# Visualizamos la tabla de resultados
kable(x = modelos, caption = "Comparativa de la precisión de los distintos modelos",
      col.names = c("Sin regularización", "Con regularización"))
```

Table 1: Comparativa de la precisión de los distintos modelos

	Sin regularización	Con regularización
(1) PC1 + PC2 + PC3 + PC11	0.8929540	0.8924175
(2) PC1 + PC2 + PC3 + PC4 + PC11	0.9017037	0.9013212
(3) PC1 + PC2 + PC3 + PC4 + PC8 + PC11	0.9019789	0.8999665
(4) PC1 + PC2 + PC3 + PC4 + PC6 + PC8 + PC11	0.9056853	0.8993535
(5) PC1 + PC2 + PC3 + PC4 + PC6 + PC8 + PC11 + PC19	0.9042058	0.9032249

Tras comprobobar las precisiones de los distintos modelos de regresión logística probados con y sin regularización podemos observar que, las diferencias entre ellas no son realmente grandes. Los beneficios obtenidos por la regularización son mínimos y, en principio, consideramos un buen equilibrio entre complejidad y calidad escoger el modelo número 4 sin regularización.

```
# Nos quedamos con el modelo 4
 etiquetas = predict(modelo4, newdata = training)
 ein = sum(etiquetas != spam[i$training, ncol(spam)])*100/length(i$training)
 etiquetas = predict(modelo4, newdata = test)
 eout = sum(etiquetas != spam[i$test, ncol(spam)])*100/length(i$test)

 cat("El error dentro de la muestra de entrenamiento (Ein) es de", ein, "%\n")
```

```
## El error dentro de la muestra de entrenamiento (Ein) es de 9.61852 %
```

```
cat("La estimación del error fuera de la muestra (Eout) es de", eout, "%\n")
```

```
## La estimación del error fuera de la muestra (Eout) es de 9.452412 %
```

Como medida del error usamos el error de clasificación, es decir, el número de elementos mal clasificados partido de el número total de elementos en la muestra.

Con un error menor al 10 % parece que el problema que estamos tratando es lineal. Los modelos lineales probados con regresión logística, en general producen buenos resultados, si consideramos que existe algún tipo de ruido en ellos. La poca complejidad del clasificador usado ha dado lugar a que no sea muy beneficioso el uso de ningún tipo de regularización.

A lo largo del desarrollo de este problema hemos probado distintos modelos con más y menos componentes (como se puede observar en la tabla anterior), observando que la mayoría de las veces, con un mayor número de componentes el error disminuye, pero la diferencia es tan escasa que no resulta provechoso obtener un modelo más complejo para una diferencia apenas notable, por lo que buscando un equilibrio entre complejidad-resultados, hemos escogido el modelo 4.

Por último, añadir que en la página proporcionada de la que son extraídos el problema y los datos, afirman que el mejor error de clasificación es de, aproximadamente, un 7%, por lo que podemos afirmar que nuestro clasificador da unos resultados bastantes buenos sin uso de una gran complejidad.

REGRESION - LOS ANGELES OZONE

Procedemos a leer los datos del problema de regresión **Los Angeles Ozone**, en el cual intentamos predecir la concentración de ozono en la ciudad basándonos en ocho medidas meteorológicas.

Procedemos inicialmente a la lectura de datos excluyendo de las muestras aquellos que tengan alguna característica con un valor perdido. Además, quitamos el atributo día del año, que no nos sirve como predictor.

```
# Establecemos la semilla aleatoria
set.seed(11)

# Cambiamos de directorio
setwd("./datos")

# Lectura del fichero
LAozone = read.table("LAozone.data", sep=",",head=T)

# Salimos del directorio de los datos
setwd("..")

# Eliminación de valores perdidos
LAozone = na.omit(LAozone)

# Eliminamos el atributo día del año al no ser representativo
LAozone = LAozone[,-10]

# Realizamos la partición de los subconjuntos
i = particionar(data = LAozone)
```

Procedemos a preprocesar los datos:

- Eliminamos las variables con un grado de correlación superior al 90%.
- Utilizamos *BoxCox* para transformar los atributos asimétricos reduciendo así la sensibilidad a la distancia entre ellos.

```
# 1. ELIMINAMOS LAS VARIABLES CON UN ALTO GRADO DE CORRELACIÓN.
# Obtenemos la matriz de correlaciones de los datos de training
descrCor = cor(LAozone[i$training,])

# Cogemos aquellos atributos cuyo grado de correlación es mayor al 90%
highCor = findCorrelation(descrCor, cutoff = .90)
# Quitamos dichos atributos
if (length(highCor) != 0) spam = spam[,-highCor]

# 2. REALIZAMOS EL PREPROCESAMIENTO CON LAS TRANSFORMACIONES
# PREVIAMENTE MENCIONADAS Y EL BOXCOX
# Calculamos la transformación del preprocesamiento basándonos en training
preProc = preProcess(x = LAozone[i$training,2:ncol(LAozone)],
                     method = c("BoxCox", "center", "scale"),
                     outcome = LAozone[i$training, 1],
                     )

# Aplicamos la transformación a los subconjuntos.
training = predict(preProc, LAozone[i$training,2:ncol(LAozone)])
test = predict(preProc, LAozone[i$test,2:ncol(LAozone)])

# Transformamos los subconjuntos en dataframes
training = data.frame(training)
test = data.frame(test)
test["y"] = LAozone[i$test,1]
```

Llegados a este punto, vamos a utilizar la función *regsubsets*, la cual se encarga de ir buscando las combinaciones de variables (ya preprocesadas) más significativas de las disponibles en el conjunto de datos.

A continuación aplicamos regresión lineal (gaussiana) con y sin regularización, obteniendo el error de cada una aplicando *bootstrapping* para obtener 25 muestras de entrenamiento por modelo y seleccionando aquel con mayor bonanza definida por el estadístico R^2 (*R-squared*).

```
# Realizamos la selección de modelos
formulas = regsubsets(x= training, y = LAozone[i$training, 1])
# Mostramos la tabla de modelos a probar
summary(formulas)$outmat
```

```
##          vh  wind humidity temp ibh dpg ibt vis
## 1  ( 1 ) " " " "  " "      "*"  " " " " " " " "
## 2  ( 1 ) " " " "  " "      " "  " " "*" "*" " "
## 3  ( 1 ) " " " "  "*"      "*"  "*" " " " " " "
## 4  ( 1 ) " " " "  "*"      "*"  "*" " " " " "*"
## 5  ( 1 ) " " "*"  "*"      "*"  "*" " " " " "*"
## 6  ( 1 ) " " " "  "*"      "*"  "*" "*" "*" "*"
## 7  ( 1 ) "*" " "  "*"      "*"  "*" "*" "*" "*"
## 8  ( 1 ) "*" "*"  "*"      "*"  "*" "*" "*" "*"
```

```
# Pasamos la matriz preprocesada a un dataframe
training_df = data.frame(training)
training_df["y"] = LAozone[i$training,1]
```

```r
# Creamos la matriz de rsquared
modelos = matrix(nrow = 6, ncol = 2)

# Función que dado un modelo ajustado devuelve el Rsquared para la mejor
# combinación de alfa y lambda
obtenerRSquared = function(modelo) {
  modelo$results$Rsquared[modelo$results$lambda == modelo$bestTune$lambda &
                          modelo$results$alpha  == modelo$bestTune$alpha]
}

# Ajustamos todos los modelos

modelo1 = train(y ~ humidity + temp + ibh, data = training_df, method="glm",
                family="gaussian")
modelo1reg = train(y ~ humidity + temp + ibh, data = training_df,
                   method="glmnet", family="gaussian")

modelo2 = train(y ~ humidity + temp + ibh + vis, data = training_df, method="glm",
                family="gaussian")
modelo2reg = train(y ~ humidity + temp + ibh + vis, data = training_df,
                   method="glmnet", family="gaussian")

modelo3 = train(y ~ wind + humidity + temp + ibh + vis, data = training_df, method="glm",
                family="gaussian")
modelo3reg = train(y ~ wind + humidity + temp + ibh + vis,
                   data = training_df, method="glmnet", family="gaussian")

modelo4 = train(y ~ humidity + temp + ibh + dpg + ibt + vis,
                data = training_df, method="glm", family="gaussian")
modelo4reg = train(y ~ humidity + temp + ibh + dpg + ibt + vis,
                   data = training_df, method="glmnet", family="gaussian")

modelo5 = train(y ~ vh + humidity + temp + ibh + dpg + ibt + vis,
                data = training_df, method="glm", family="gaussian")
modelo5reg = train(y ~ vh + humidity + temp + ibh + dpg + ibt + vis,
                   data = training_df, method="glmnet", family="gaussian")

modelo6 = train(y ~ vh + wind + humidity + temp + ibh + dpg + ibt + vis,
                data = training_df, method="glm", family="gaussian")
modelo6reg = train(y ~ vh + wind + humidity + temp + ibh + dpg + ibt + vis,
                   data = training_df, method="glmnet", family="gaussian")

# Creamos la matriz de medidas Rsquared

modelos[1,1] = modelo1$results$Rsquared
modelos[1,2] = obtenerRSquared(modelo1reg)
modelos[2,1] = modelo2$results$Rsquared
modelos[2,2] = obtenerRSquared(modelo2reg)
modelos[3,1] = modelo3$results$Rsquared
modelos[3,2] = obtenerRSquared(modelo3reg)
modelos[4,1] = modelo4$results$Rsquared
modelos[4,2] = obtenerRSquared(modelo4reg)
modelos[5,1] = modelo5$results$Rsquared
```

```
modelos[5,2] = obtenerRSquared(modelo5reg)
modelos[6,1] = modelo6$results$Rsquared
modelos[6,2] = obtenerRSquared(modelo6reg)

rownames(modelos) = c("(1) humidity + temp + ibh",
                      "(2) humidity + temp + ibh + vis",
                      "(3) wind + humidity + temp + ibh + vis",
                      "(4) humidity + temp + ibh + dpg + ibt + vis",
                      "(5) vh + humidity + temp + ibh + dpg + ibt + vis",
                      "(6) vh + wind + humidity + temp + ibh + dpg + ibt + vis")

kable(x = modelos, caption = "Comparativa del RSquared de los distintos modelos",
      col.names = c("Sin regularización", "Con regularización"))
```

Table 2: Comparativa del RSquared de los distintos modelos

	Sin regularización	Con regularización
(1) humidity + temp + ibh	0.6879886	0.6668323
(2) humidity + temp + ibh + vis	0.6833931	0.6726951
(3) wind + humidity + temp + ibh + vis	0.6790673	0.6794456
(4) humidity + temp + ibh + dpg + ibt + vis	0.6696048	0.6695554
(5) vh + humidity + temp + ibh + dpg + ibt + vis	0.6620325	0.6602524
(6) vh + wind + humidity + temp + ibh + dpg + ibt + vis	0.6660743	0.6687061

Tras comprobar los R-squared de los distintos modelos de regresión lineal (gaussiana) probados con y sin regularización podemos observar que, las diferencias entre ellas no son especialmente significativas. No obstante, cabe destacar, que el mejor modelo que hemos obtenido ha sido el más simple (menor número de variables) y sin regularización, por tanto, procedemos a seleccionar el modelo 1 como modelo final.

```
# Función que calcula el error porcentual en regresión, dados dos conjuntos
# de etiquetas
error_porcentual = function(etiquetas_test, etiquetas_pesos){
  diferencia = max(c(etiquetas_pesos, etiquetas_test))
               -min(c(etiquetas_pesos, etiquetas_test))
  error = mean(abs(etiquetas_test - etiquetas_pesos)/diferencia)
  error
}
# Nos quedamos con el modelo 4
 etiquetas = predict(modelo1, newdata = training)
 ein = error_porcentual(LAozone[i$training,1], etiquetas)
 etiquetas = predict(modelo1, newdata = test)
 eout = error_porcentual(LAozone[i$test,1], etiquetas)

 cat("El error dentro de la muestra de entrenamiento (Ein) es de", ein*100, "%\n")
```

```
## El error dentro de la muestra de entrenamiento (Ein) es de 10.66151 %
```

```
 cat("La estimación del error fuera de la muestra (Eout) es de", eout*100, "%\n")
```

```
## La estimación del error fuera de la muestra (Eout) es de 9.821542 %
```

Como medida del error porcentual utilizamos la diferencia en valor absoluto de la cantidad de ozono predicha con respecto a la original dividido por la diferencia entre la máxima cantidad y mínima cantidad de ozono disponible.

Con un error alrededor 10 % parece que el problema que estamos tratando es otra vez lineal. Los modelos lineales probados con regresión lineal, en general producen buenos resultados, si consideramos que existe algún tipo de ruido en ellos. Además, hemos podido observar que obtenemos mejores resultados incluso sin utilizar regularización, ya que el problema no es muy complejo.

A lo largo del desarrollo de este problema hemos probado distintos modelos de entre 3 y 7 componentes, observando que conforme aumenta el número de componentes, por regla general, aumenta el error en la muestra.

No conocemos el mejor error aproximado que podemos obtener en este problema. No obstante, el error cercano al 10 % nos hace creer que con los métodos lineales disponibles hemos hecho un buen ajuste.