**Ejemplos de Regresión Lineal en R. Regresión Lineal Simple**

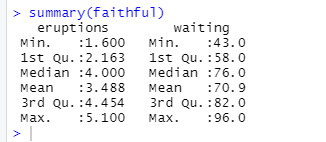
Vamos a utilizar el dataset faithful13 que relaciona la duración de una erupción de un Geisser en el parque de Yellowstone en USA con el tiempo que transcurre entre una erupción y la siguiente y que está disponible directamente en R por encontrarse en el paquete datasets14

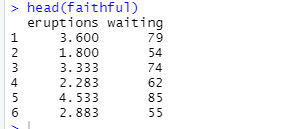
Cargamos el dataset faithful y las librerías necesarias para alguno de los algoritmos de regresión lineal





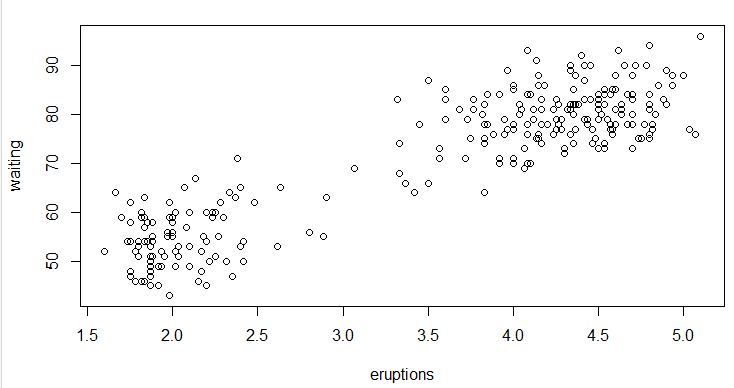
Hacemos un summary() y un head() para ver su contenido



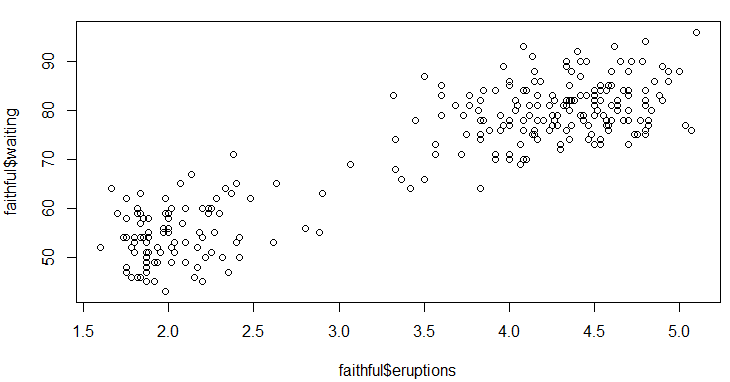


Representamos gráficamente la relación entre las dos variables (Figura 3): eruptions (duration) vs waiting (tiempo de espera entre erupciones







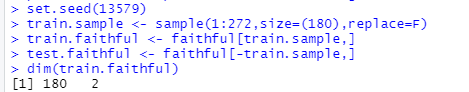


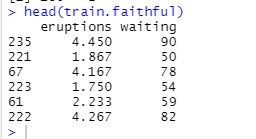
Veamos con la función cor() la correlación entre ambas variables. Nos dará una pista de cuan bueno va a ser el predictor, cuanto más se acerque a la unidad querrá decir que más relacionadas están las variables y más fácil resultará ajustar una buena predicción



La correlación es bastante alta en este caso.

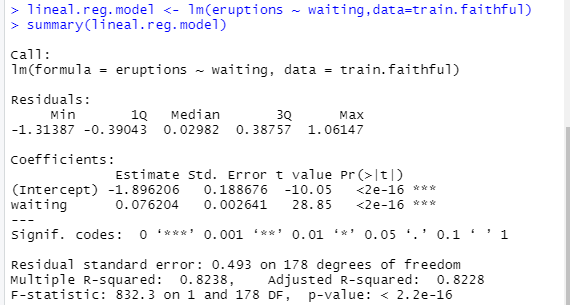
A partir del dataset original vamos a crear un conjunto de entrenamiento para que se genere el modelo lineal a partir del mismo y otro de test para ver como de bien es capaz de predecir sobre otro conjunto de datos diferente al de entrenamiento



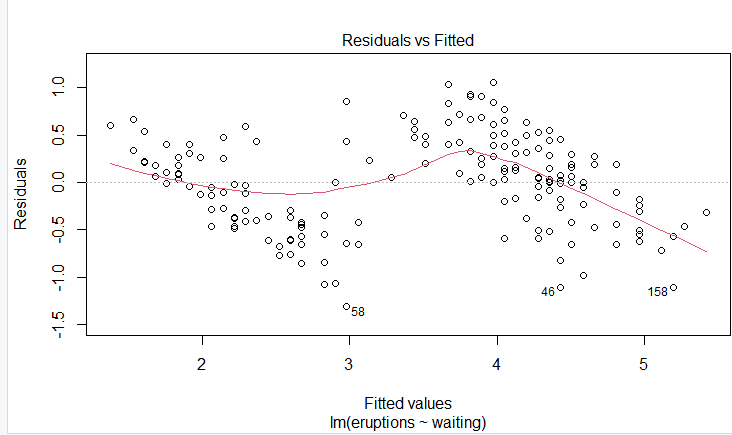


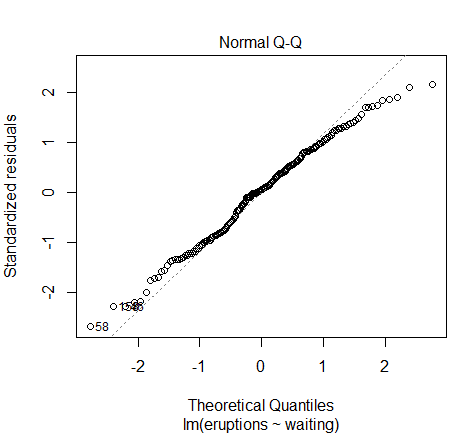
Podemos observar que tenemos dos datasets (para entrenamiento y test) de las mismas características que el dataset original

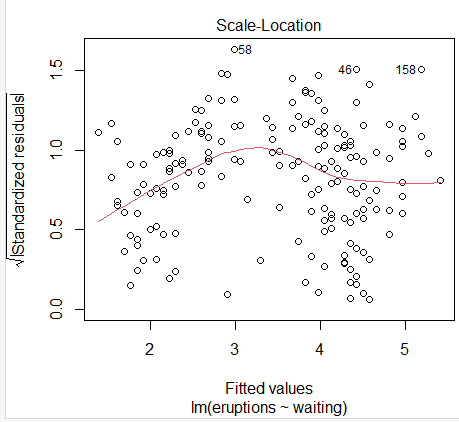
Entrenamos el modelo con la función lm() con la que se implementa el algoritmo Ordinary Least Squares Regression como ya hemos comentado anteriormente

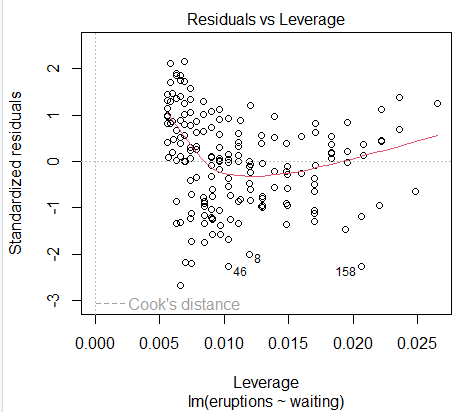




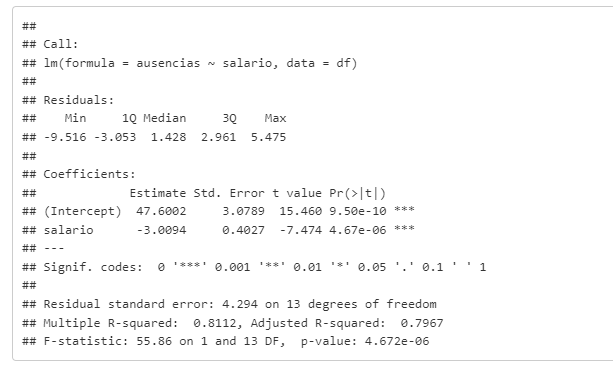








**Interpretación Ejemplo**

****

El modelo ajustado es significativo: Los **coeficientes de regresión son 47.6002 y -3.0094**, estos parámetros son significativos, con p-valor menor de 0.05 (9.50e-10, 4.67e-06). **El error estándar para cada parámetro es 3.0789 y 0.4027** respectivamente. La **R2**

**ajustada es 0.7967,** que indica un buen ajuste del modelo (próximo a 1),

El modelo tiene la forma de:

ausencias=47.60−3.01×salario

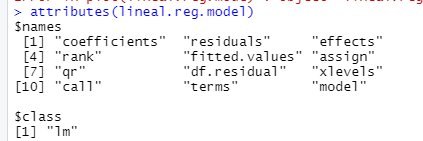
Los coeficientes de regresión son 47.60 y -3.01.

47.60 es el valor medio de la variable dependiente (ausencias) cuando la predictora es cero (salario).

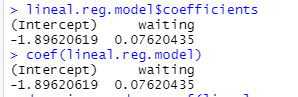
3.01 es el efecto medio (negativo) sobre la variable dependiente Y (ausencias) al aumentar en una unidad el valor de la predictora X (salario). Esto es, variación que se produce en Y (-3.01) por cada unidad de incremento en X.

EN CONCLUSIÓN: Existe una relación lineal negativa entre las variables: cuando aumentamos en una unidad el salario, las ausencias disminuyen en 3.01 unidades. De forma que por cada aumento de la categoría del salario, las ausencias de los trabajadores disminuyen en 3.01 días.

Podemos obtener más detalles del modelo utilizando la función attributes()



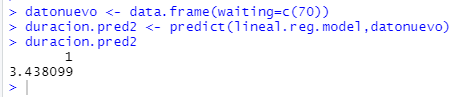
Vamos a ver de nuevo el valor de los coeficientes utilizando el atributo $coefficients y la función coef()



Si quisiéramos predecir el valor de la duración de la erupción para un nuevo valor de tiempo entre erupciones podríamos montar la ecuación con los coeficientes y aplicarla a ese nuevo valor. Supongamos que tenemos un tiempo entre erupciones de 70 minutos



Pero hay una forma mucho más elegante de obtener una predicción y es usar la función predict() #Creamos un dataset con sólo la variable predictora (waiting)



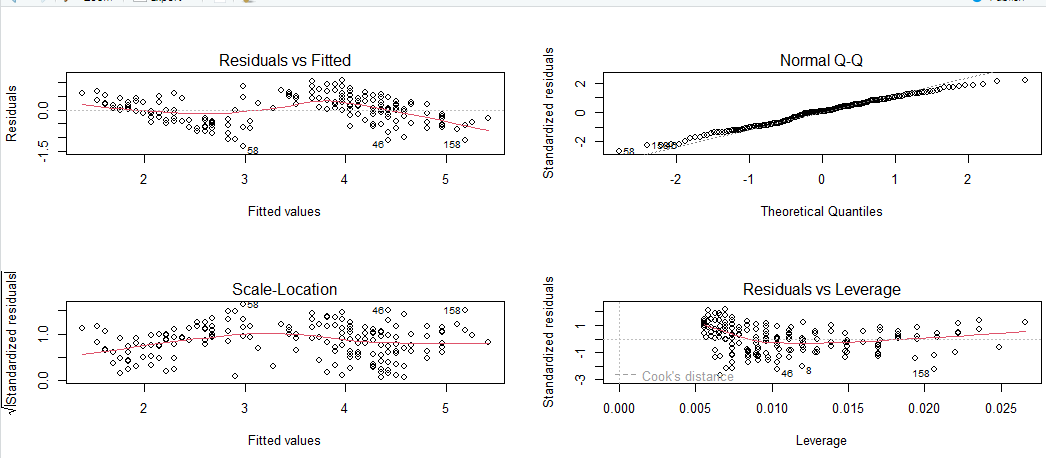
Por último vamos a comparar el RSE (Residual Standard Error) en el dataset de entrenamiento y en el de test para compararlos. Se podría hacer con la función rmse() del paquete Metrics, pero lo vamos a hacer directamente aplicando la fórmula que hemos visto a ambos datasets



*# En realidad fitted - la erupcion real de las observaciones son los residuos*

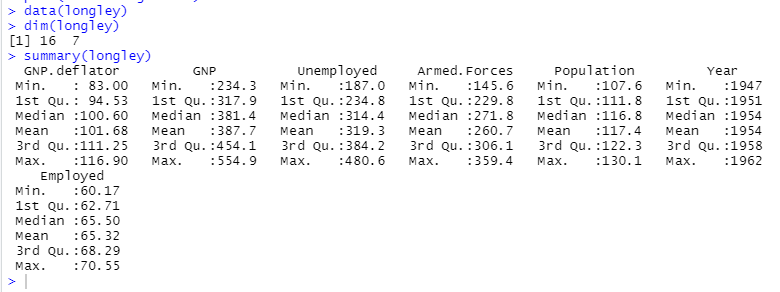




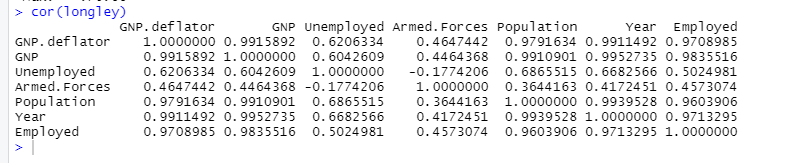


**Regresión Lineal Multivariable**

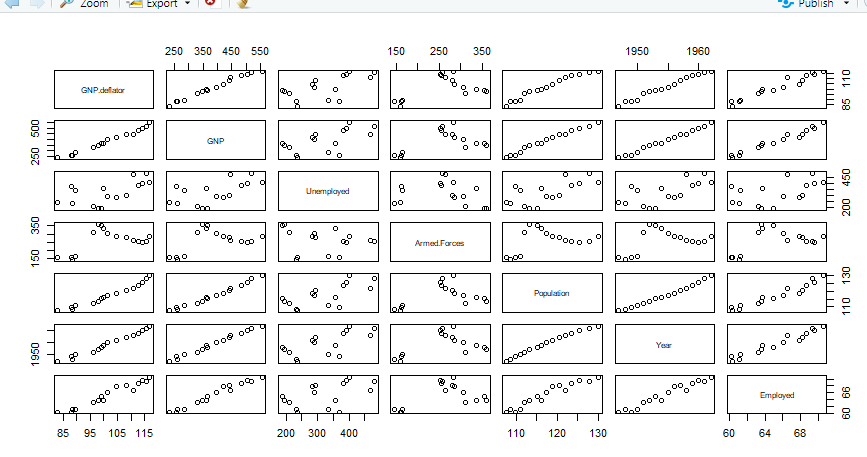
Para el ejemplo de regresión multivariable seleccionamos el paquete longley15 con 7 variables económicas y se trata de predecir el número de empleados en base a dichas variables. Son todas variables de tipo numérico



Vemos la correlación entre variables con la función cor() y su representación gráfica con pairs()

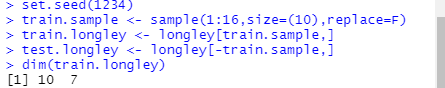


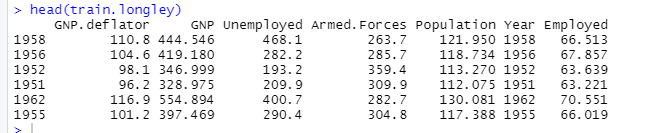




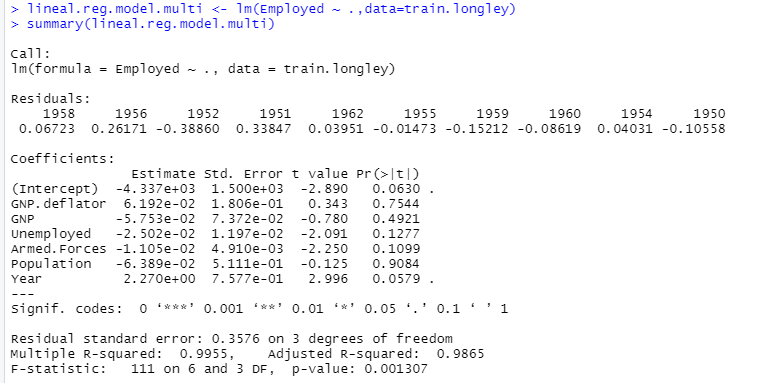
Tanto en la tabla como en el gráfico se ve que el número de empleados (Employed) tiene una correlación bastante alta con las variables Year, Population, GNP y GNP.deflator. En las gráficas se puede observar perfectamente que en esos casos los puntos parecen seguir prácticamente una línea recta

Creamos el conjunto de entrenamiento y de test a partir del dataset original





Entrenamos el modelo con la función lm() (algoritmo Ordinary Least Squares Regression) para obtener los coeficientes de la regresión lineal



Nótese que en la función lm(), en vez de escribir todas las variables de entrada (GNP, Unemployed ...) hemos colocado un punto (.) que simboliza "todas las variables"

Los coeficientes resultantes para este modelo a partir de los datos de entrenamiento son **B0 = -6.306e+03, B1 = 1.978e-02, B2 = -1.374e-01, B3 = -3.717e-02, B4 = -1.307e-02, B5 = 5.709e-01, B6 = 3.259e+00. La notación e+ o e- es equivalente a 10 elevado a la... es decir e-02 = 10-2 = 0,01**

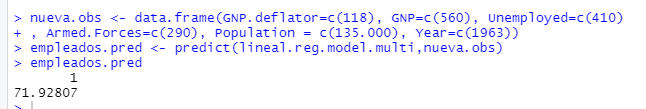
Y la fórmula de regresión completa por tanto:

**Employed = -6.306e+03 + 1.978e-02 \* GNP.deflactor -1.374e-01 \* GNP -3.717e-02 \* Unemployed -1.307e-02 \* Armed.Forces + 5.709e-01 Population + 3.259e+00 \* Year**

Podemos observar que los dos coeficientes mayores y por tanto que más peso e influencia tendrán en el resultado final son la población (Population) y el año en el que nos encontremos (Year)

Por último obtenemos una predicción usando la función predict(), por lo que tenemos que crear un dataframe con una observación sobre la que queramos hacer la predicción y que contenga todas las variables. En este caso vamos a intentar la predicción para el año 1963 que es el primero que no aparece en la serie de observaciones con valores inventados para las variables de entrada, pero con cierto sentido siguiendo las tendencias de los años anteriores:

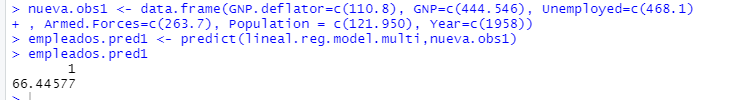
#Creamos un dataset con las variables predictoras



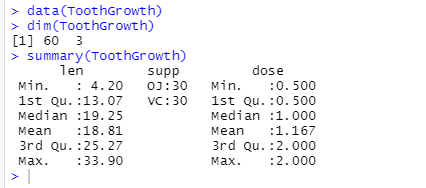
El resultado que nos da: Employed = 71.92807 sería la predicción de empleabilidad para el año 1963. Una potente herramienta sin duda y que funciona bien siempre y cuando alguna de las variables influyentes y que no estén contempladas en el modelo cambien alterándolo drásticamente

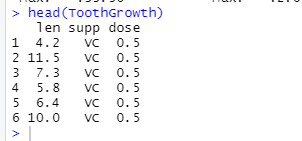
Vamos a intentar la predicción para el año 1958



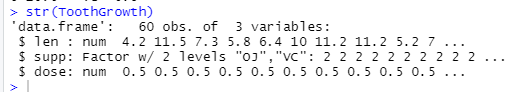


Hasta el momento sólamente hemos utilizado datasets con variables numéricas, pero ¿qué ocurre si entre los predictores hay variables categóricas? Vamos a utilizar para ver esto el dataset ToothGrowth16 que observa el crecimiento de los dientes de las cobayas utilizando un determinado tipo de suplemento y distintas dosis de dicho suplemento





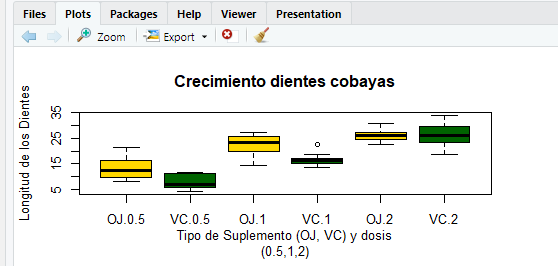
Utilizando la función **str()** podemos ver la estructura de los campos del dataset ToohGrowth



Vemos en este caso que la variable supp (suplemento) es una variable categórica (factor) con dos posibles valores (leves) que son OJ y VC

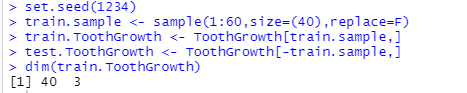
Utilizando la función boxplot() podemos ver una representación gráfica muy clara de lo que influye cada variable predictora en el crecimiento de los dientes que es la variable objetivo (target)

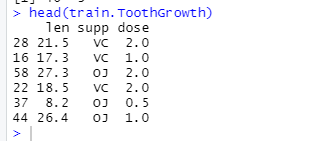




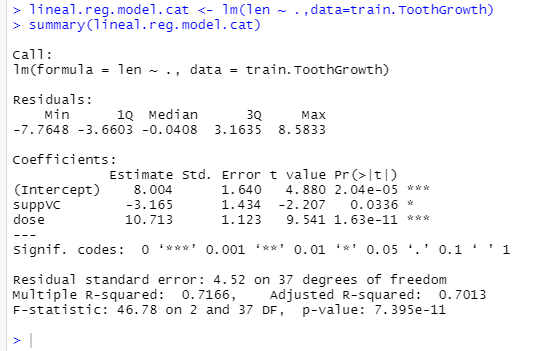
En la Figura 7 se puede ver que a misma dosis de suplemento, las cobayas que toman el de tipo OJ (amarillo) experimentan un mayor crecimiento en sus dientes, sobre todo cuando están tomando dosis más pequeñas

Como en los ejemplos anteriores creamos el conjunto de entrenamiento y test a partir del dataset original





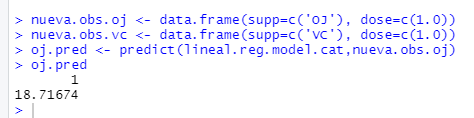
Y construimos el modelo con **lm()**

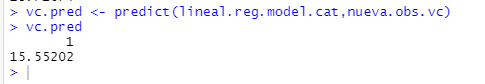


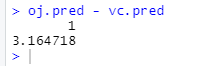
Aquí vemos la principal diferencia con los modelos anteriores y es que en el segundo coeficiente (B1) aparece el nombre de la variable (supp) concatenado con uno de sus posibles valores (vc). La forma de interpretar esto es la siguiente: cuando el suplemento sea vc se aplicará la **fórmula Y = B0 + B1 + B2 \* X2** o más concretamente en este **caso len = 8.756 -3.646 + 9.801 \* dose,** mientras que cuando el suplemento sea oj no se aplica el valor de B1, quedando **Y = B0 + B2X2 (len = 8.756 + 9.801 \* dose).**

Realicemos la predicción (**predict()**) con ambos suplementos para una misma dosis y así ver la diferencia

#Creamos dos datasets con las variables predictoras (uno por cada suplemento)





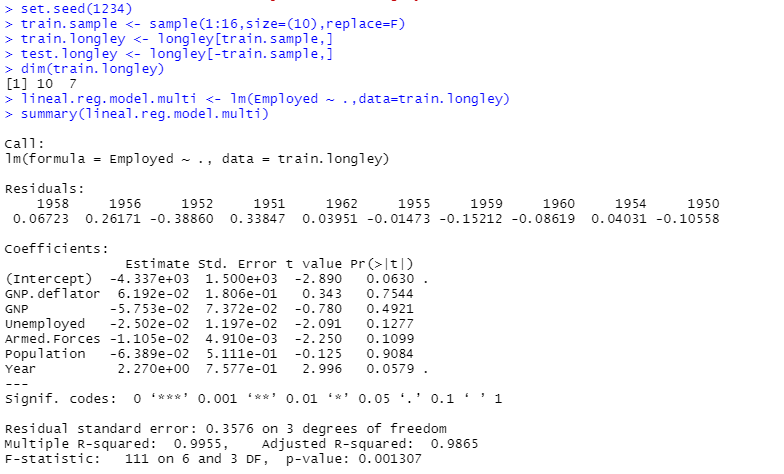


Como preveíamos, después de observar la gráfica de la Figura 7 el crecimiento con el suplemento OJ es mayor que con el suplemento VC con la misma dosis (1.0). La diferencia justo es el valor del coeficiente suppvc que aplicamos en la fórmula de regresión cuando la variable supp=VC y que no aplicamos cuando supp=OJ.

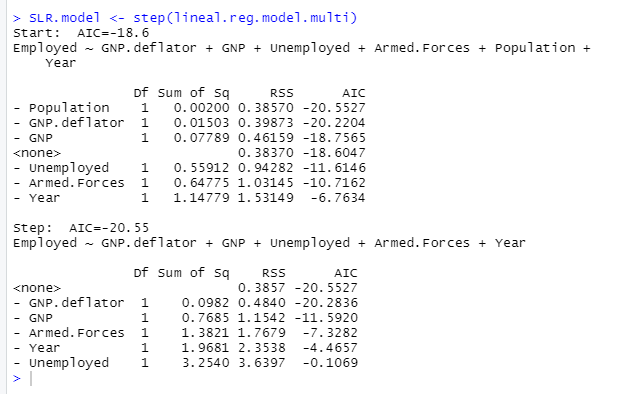
**Otros algoritmos de regresión lineal**

**Stepwize Linear Regression. # modelo base lineal.reg.model.multi**

**#aplicamos stepwise al modelo base**

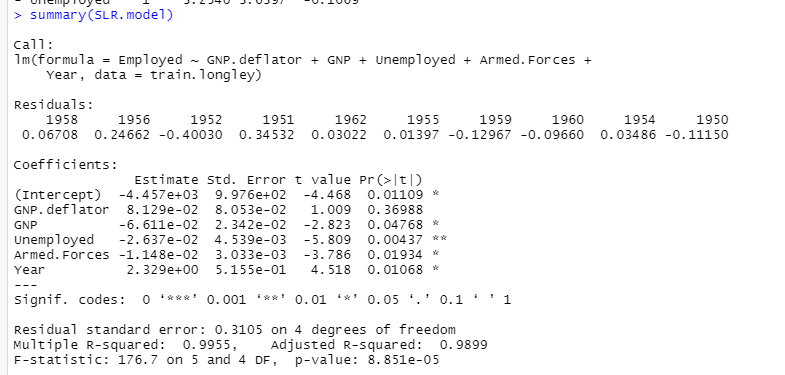
****

**# ejecutamos step-wise para seleccionar las variables predictoras**

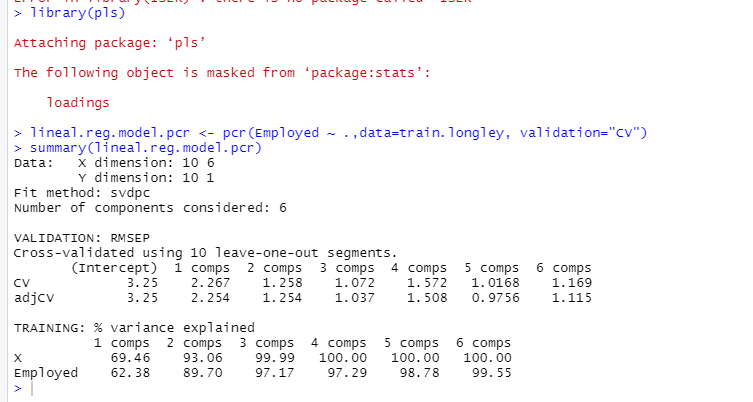
****

**# GNP.deflator desaparece de las variables predictoras después de aplicar stepwise**

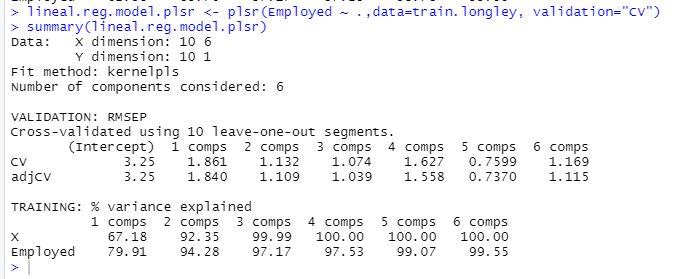
**# summarize the selected model**

****

**Principal Component Regression**

****

**Partial Least Squares Regression**

****

**Regresión Lineal Penalizada**

Este tipo de regresiones busca ajustar un modelo, no eliminando algunos de los predictores, como hemos visto anteriormente, si no usando técnicas que fuerzan a los coeficientes hacia valores nulos aunque no es tarea fácil en muchos casos

Los principales paquetes de R para poder ejecutar los modelos de Regresión Lineal penalizada son glmnet() y Lars() , más en concreto:

**Regresión no Lineal**

Los modelos de regresión no lineal tratan de generar una ecuación que describe una relación no lineal entre una variable objetivo continua y una o más variables predictoras. Se usa por tanto esta técnica cuando no se pueda aplicar un modelo óptimo sólo con parámetros lineales. Se dice que los parámetros son lineales cuando cada término en el modelo es aditivo y contiene solamente parámetros que multiplican los términos.

En estos casos se debe especificar la que se conoce como “función de expectativas” (expectation function) que se emplea para ejecutar la regresión no lineal. La elección de esta función a menudo depende del conocimiento de la forma de la curva de respuesta o el comportamiento de las propiedades del sistema pero suelen incluir formas cóncavas, convexas, exponenciales creciente o decrecientes, sigmoidales (S) y asintóticas

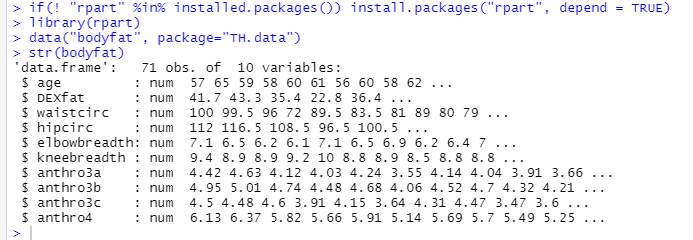
**Regresión no lineal con Árboles de Decisión**

**Ejemplos de la regresión no lineal por Árboles de Decisión en R**

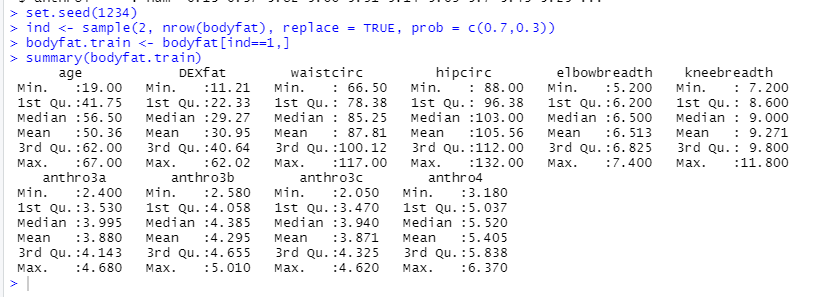
**Árboles de regresión con CART**

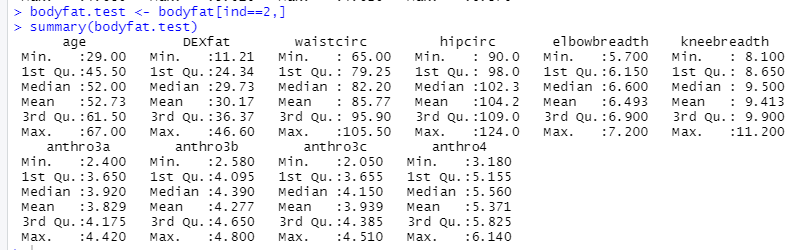
Cargamos las librerías necesarias para ejecutar Árboles de Decisión para regresión con CART

Cargamos en este caso el dataset bodyfat del paquete TH.data y vemos la estructura del dataset



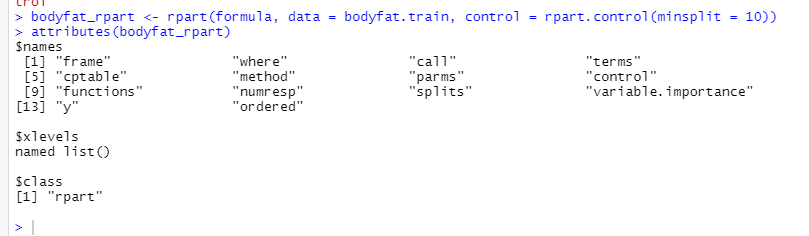
# Generamos los dataset de entrenamiento y test

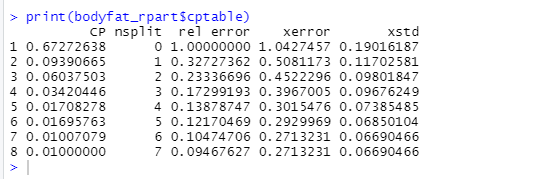


****

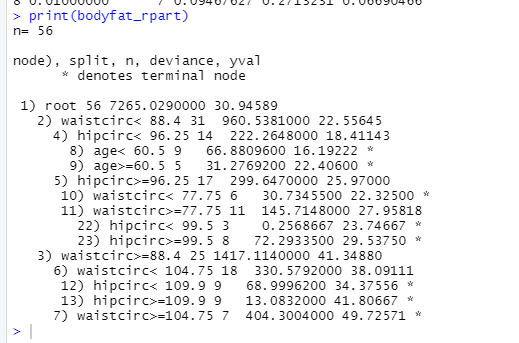
**# Se entrena el modelo con rpart()**

****

****

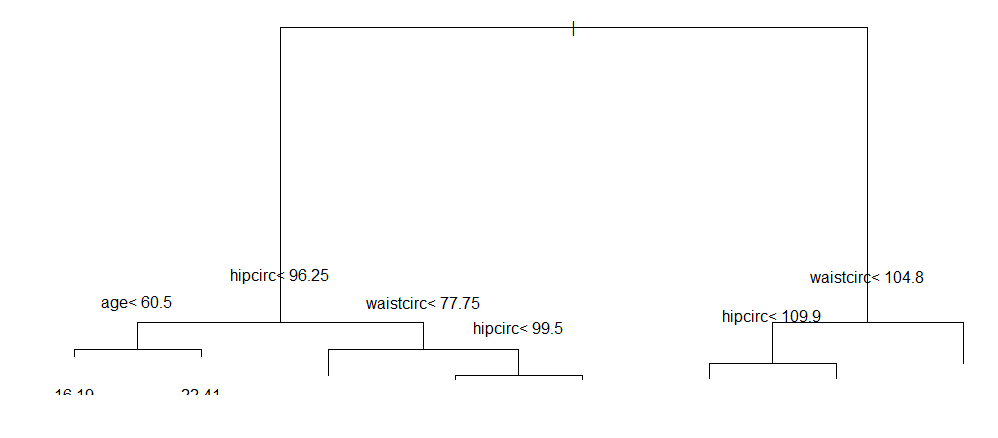
****

**# Vemos las reglas generadas con la función rpart()**

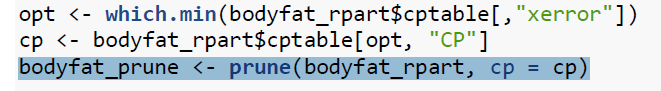
****

**# Mostramos la representación gráfica del árbol con la función plot()**

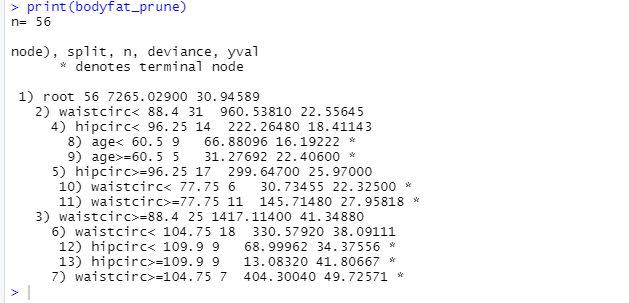
****

****

**# Podamos el árbol y obtenemos aquel con el mínimo error de predicción**

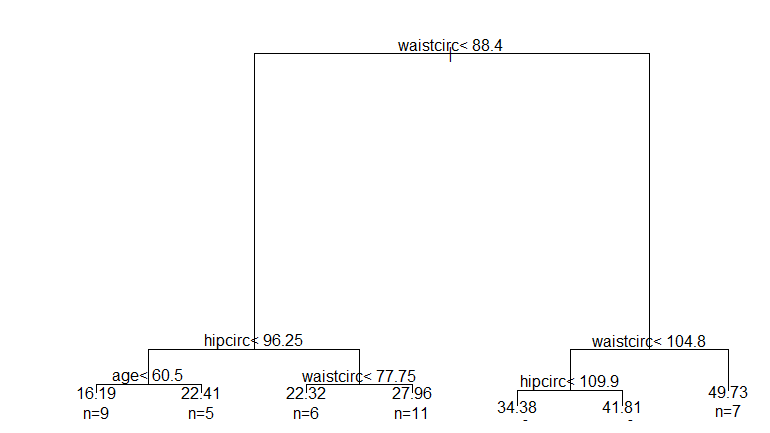
****

**# Vemos como quedan las reglas del árbol podado**

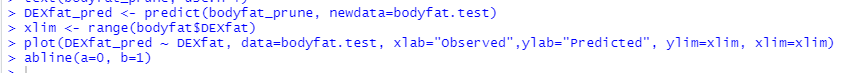
****

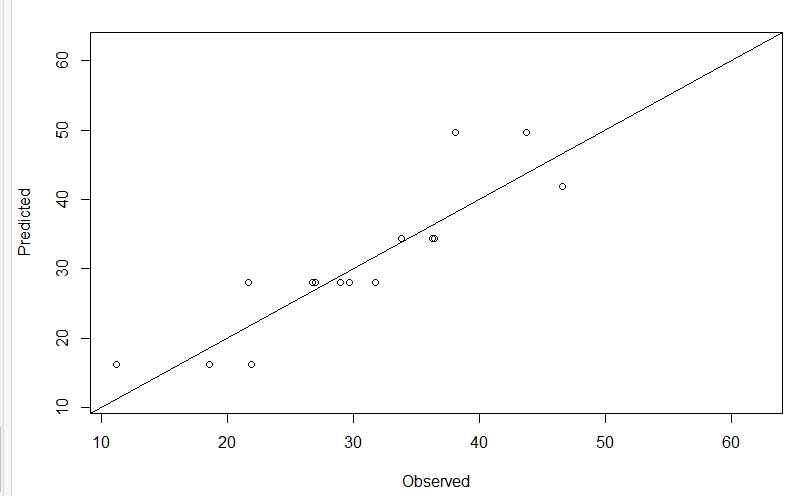
**# Representamos gráficamente el árbol podado**

****

****

**# Predecimos los nuevos valores con predict()**

****

****

**Algoritmos de clasificación**

**Clasificación Lineal**

**Regresión Logística (Logistic Regression)**

La Regresión Logística es un método de clasificación que modela la probabilidad de que una observación pertenezca a una determinada clase. Normalmente se utiliza para problemas de clasificación binario (dos únicas clases) aunque no es exclusivo pudiéndose utilizar también con más de dos clases (multinomial).

**Regresión Logística Unidimensional. Se trata de una Regresión binaria y en la que solamente participa una única variable predictora.**

Se toman dos valores discretos (por simplificación hablaremos de 0/1) y solamente existe una variable independiente X que en base a su valor hará que Y sea más probable que se acerque más al 0 o al 1.

**Regresión Logística Multivariable**

**Análisis de Discriminantes Lineales (LDA)**

**Partial Least Squares Discriminate Analysis (PLSDA)**

**Implementación de la Clasificación Lineal en R**

Los principales paquetes de R para poder ejecutar los modelos de clasificación lineal son stats52 , MASS53, caret54 , más en concreto:

• Regresión Logística en stats con la función lm()

• Análisis de Discriminantes Lineales (LDA) en MASS con la función lda()

• Partial Least Squares Discriminate Analysis (PLSDA) en caret con la función plsda()

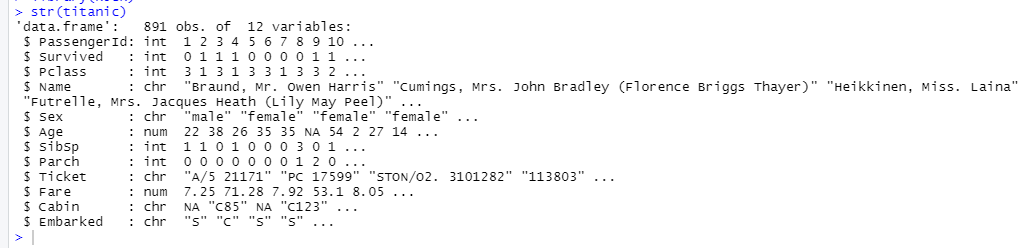
**Ejemplos de clasificación lineal en R**

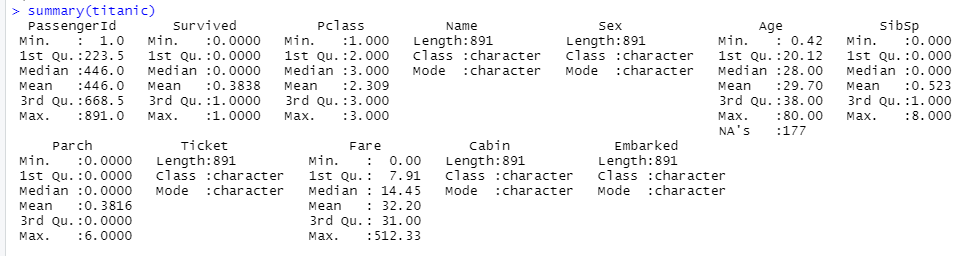
**Regresión Lineal Binaria clásica**

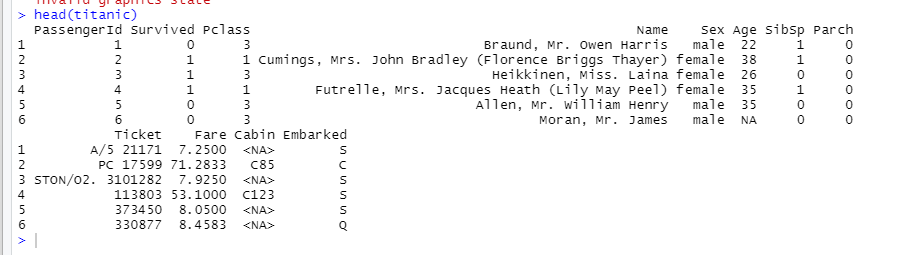
**Cargamos el dataset del Titanic (versión kaggle) 55 y algunos paquetes necesarios**

****

Hacemos str() para ver la estructura del objeto, summary() para ver la distribución de sus variables, dim() para ver las dimensiones y head() para ver el contenido del dataset

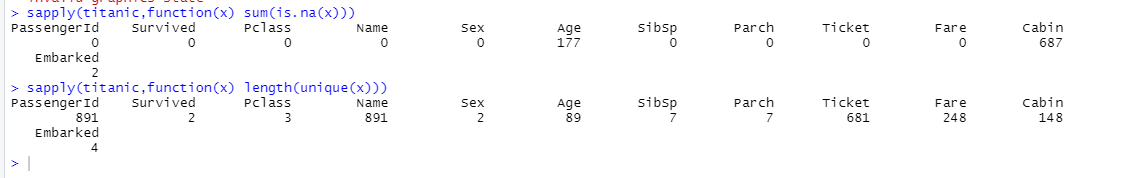








Realizamos un pequeño análisis exploratorio para ver los valores únicos de cada variable y los desconocidos **(NA**). Utilizamos para ello la función **sapply()**

****

Se puede ver que hay una observación por pasajero, así que no hay que hacer mayor transformación en ese sentido y podemos eliminar las variables PassengerId, Name, Ticket y Cabin del modelo si las usamos tal cual por ser variables categóricas con múltiples valores lo que no hace sinó aportar complejidad al modelo haciéndolo inviable. Utilizamos la función select() del paquete dplyr.

****

Para terminar de acondicionar los datos vamos a tratar los valores desconocidos (NA). Viendo los resultados del exploratorio anterior vemos que hay NA en la edad (Age) y el embarque (Embarked)

# La edad de las observaciones desconocidas la sustituimos por la media

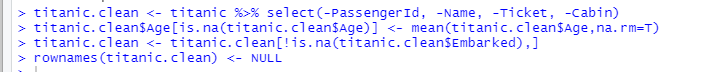


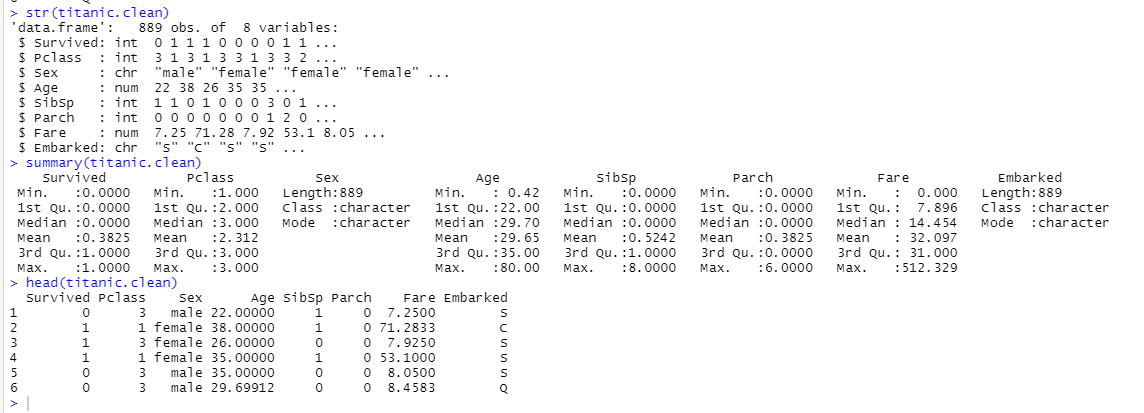
# Las observaciones con Embarked desconocido las eliminamos directamente



# Quitamos también el número de observación (rowname



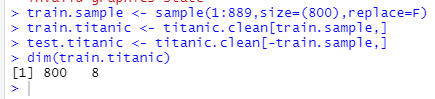


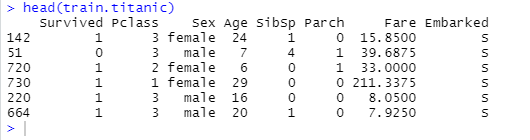


Para que la regresión funcione es necesario que las variables categóricas sean factores. Si observamos la salida que obtuvimos de la función str() vemos que las variables Sex y Embarked son factores

Creamos los conjuntos de entrenamiento y test

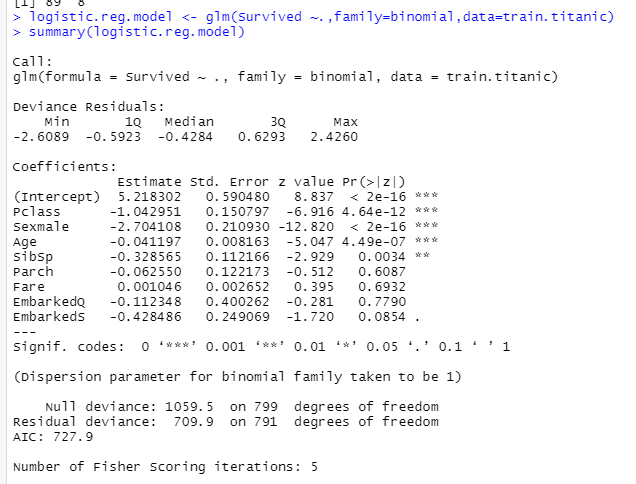






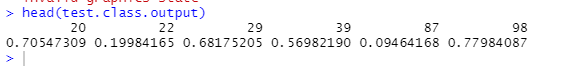


Entrenamos el modelo de regresión logística con la función glm() y aplicando el parámetro family=binomial

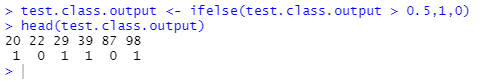


Aplicamos el modelo al conjunto de test para obtener las predicciones para esas observaciones con la función predict()





Lo que obtenemos en realidad es un vector con el porcentaje de que la clase de Supervived sea 1. Para considerar lo que es 0/1 vamos a fijar el criterio de que los que tengan el porcentaje mayor de 0.5 son unos y los que tengan menos de 0.5 son ceros



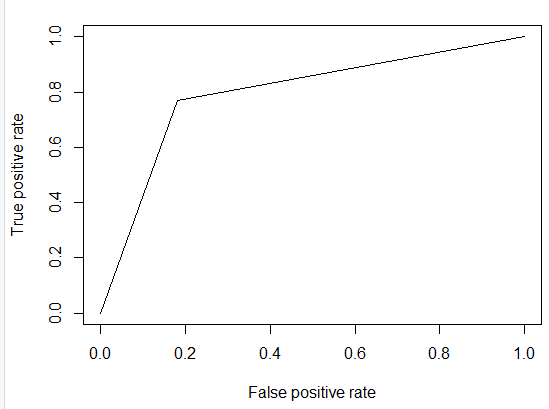
Ahora ya podemos calcular la precisión (porcentaje de aciertos) de nuestro modelo aplicándolo al conjunto de test

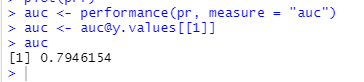




Dibujemos la **Curva ROC** cuyo significado veremos posteriormente en el apartado 0 utilizando la función **performance()**





****

**El área bajo la curva (Area Under Curve = AUC) es de 0,74. Un valor bastante bueno considerando que el máximo sería 1 (el clasificador perfecto).**

**Mas ejemplos en :** machinelearningmastery.com

**Clasificación con Árboles de Decisión**

Los árboles de decisión para clasificación (también los hay para regresión como hemos visto anteriormente) **intentan estimar la clase (valor cualitativo) a la que pertenece una observación**

Para crear el árbol se utiliza un algoritmo de partición binario que se va aplicando sobre todas las variables predictoras.

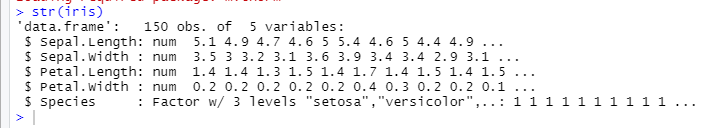
**Ejemplos de Clasificación con Árboles de Decisión en R**

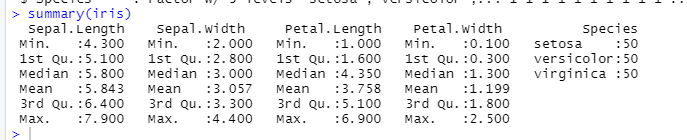
****

****

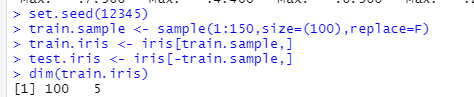
**Árboles de clasificación con PART**

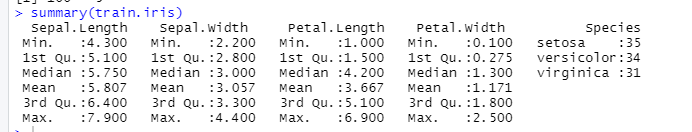
Cargamos el famoso dataset iris69 sobre el que realizaremos la clasificación multiclase. La variable target es Species y puede tomar tres valores: setosa, versicolor y virginica



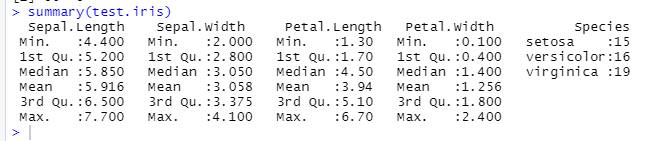


Creamos el conjunto de entrenamiento y test, como de costumbre





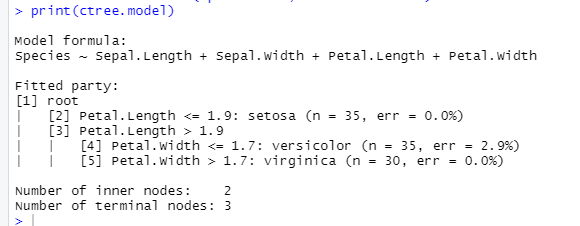




Construimos el modelo con **ctree()** del paquete party y mostramos las reglas resultado con **print()**



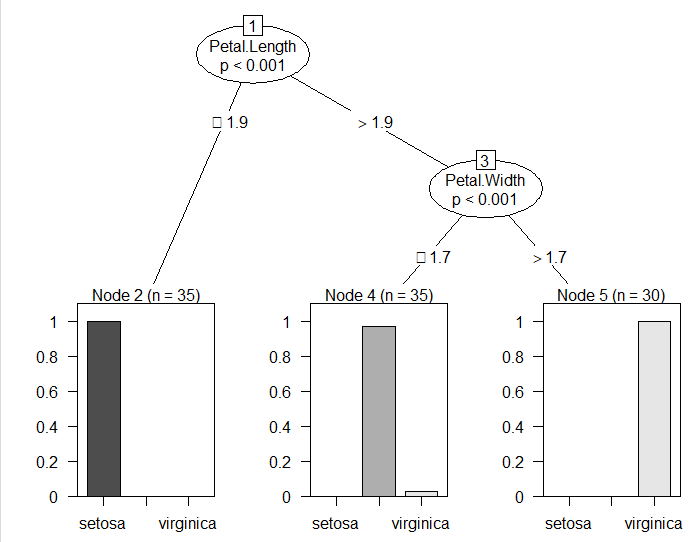
# Mostrar las reglas



Las reglas nos dicen claramente los valores de corte de las variables y el error en la clasificación en cada nodo

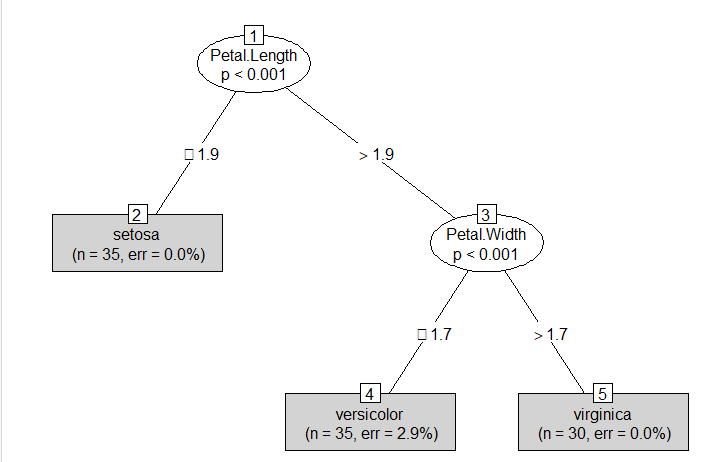
# Representar el árbol completo



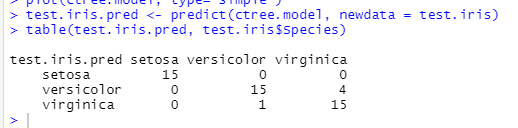


# Representar el árbol simplificado





Por último hacemos la predicción sobre el conjunto de test y medimos la precisión con la función table()



Según la tabla (se representan las columnas como el valor real y las filas como el valor de la predicción) para la clase setosa tiene una precisión del 100% acertando en las 13 observaciones disponibles en el dataset de test, para la versicolor hay 19 observaciones y acierta en 18 y para la virginica hay 18 observaciones y acierta en 16

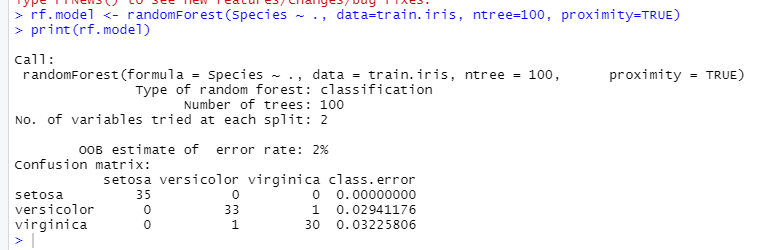
**Árboles de clasificación con Random Forest**

Cargamos las librerías necesarias para ejecutar Random Forest

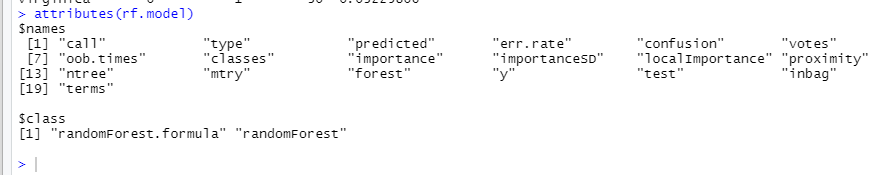


Creamos el modelo con la función **randomForest()** utilizando los mismos datos de entrenamiento y test que para el árbol simple en 0. Utilizamos un número de árboles de 100 y vemos la información del modelo con las funciones **print() y attributes()**

**# Vemos la matriz de confusion para el conjunto de entrenamiento**

****

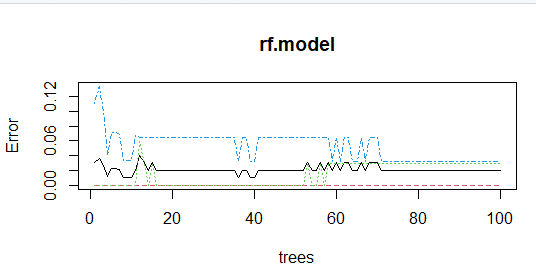
**# Vemos los atributos del modelo**

****

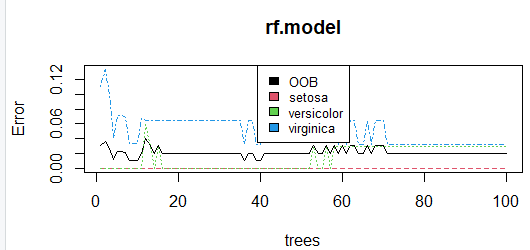
En la salida de la función print() podemos ver la matriz de confusión sobre el conjunto de entrenamiento para ver la precisión en la predicción de cada una de las clases (especies)

# plot the tree







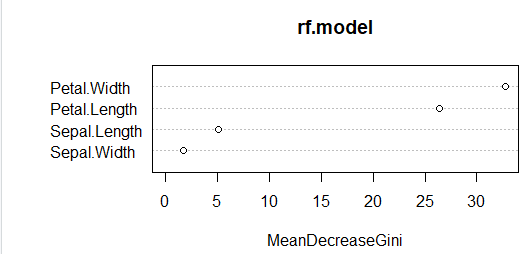


Según podemos ver en la Figura 17 la clase setosa consigue una precision máxima (error cero) con un sólo árbol como hemos visto en el ejemplo de árboles simples. Versicolor también alcanza el mínimo error con un número muy bajo de árboles, pero no desciende de un error de cerca del 0,07. Virginica padece grandes oscilaciones, se puede observar que no por utilizar más árboles el error desciende y el error tampoco desciende del 0,07 de error

Con la función varImpPlot() podemos ver una representación gráfica de la importancia de las variables en función del índice de Gini

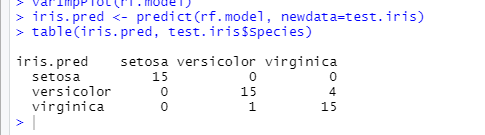
# importance of variables



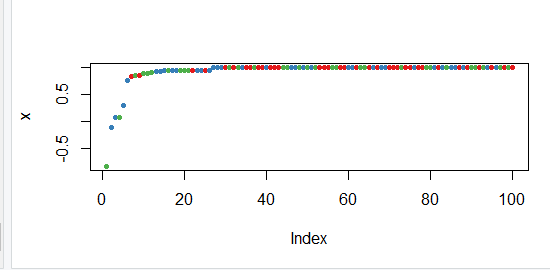


Según se puede ver en la gráfica de la Figura 188 la variable que más impacta en la construcción del modelo es la anchura del pétalo (Petal.Length) y después la longitud (Petal.Width)

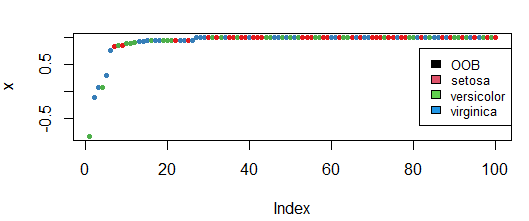
Por último realizamos la predicción sobre el conjunto de test con la función predict() y obtenemos la matriz de confusión











**Clasificación No Lineal**

**Evaluación de modelos**

Técnicas para medir la calidad de nuestros modelos.

Cómo asegurarnos de que el modelo tendrá un buen resultado ajustándose a un dataset de entrenamiento pero sin que haya un problema de sobre-entrenamiento (overfit) que le haga funcionar deficientemente con otros conjuntos de datos

Existen métricas, dependiendo del tipo de modelo, que se aplican para ver la efectividad del modelo y compararlo con otros modelos, ya sean del mismo tipo pero con distintos parámetros o de distinto tipo.

**Técnicas de remuestreo (resampling).**

Partición de datos (Data Split)

Validación Cruzada (Cross Validation):

Validación cruzada con k particiones (k-fold cross validation

Validación cruzada con k particiones repetida ( repeated k-fold cross validation)

Validación cruzada dejando uno fuera (Leave One Out Cross Validation) LOOCV.

Bootstrap

**Métricas de evaluación de modelos**

Regresión Lineal

RSE (Residual Estándar Error)

RMSE (Root Mean Square Error)

R2

F-Statistic

P-values

Regresión Logística

McFadden R2

Matriz de confusion

Curva ROC

Clasificadores binaries

Matriz de confusion

Curva ROC

Clasificadores multiclase

Precisión. Matriz de confusion

Curva ROC Multiclase

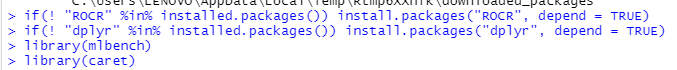
**Creación y evaluación de modelos con caret**

Caret es un paquete de R y su nombre son las iniciales de Classification and Regression Training

Para ilustrar este ejemplo vamos a utilizar el paquete de R con datasets mlbench99 y concretamente el conjunto de datos denominado Sonar que contiene una muestra de datos de un sonar con 60 variables numéricas que representan la energía en una banda de frecuencia y una variable categórica (con valor M=Mineral o R = Roca) que corresponde a la variable target que vamos a predecir.

Cargamos las librerías necesarias para ejecutar el ejemplo con caret if(!

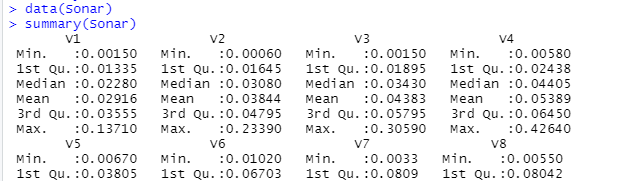


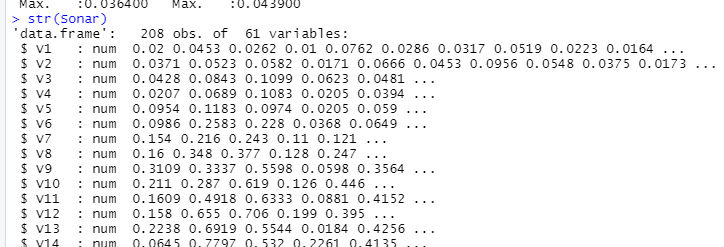


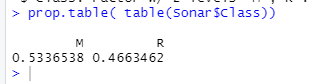




Exploración del Dataset



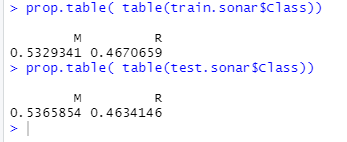




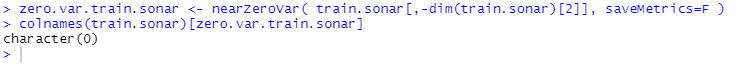
### Creamos las particiones de entrenamiento y test para los datos de Sonar



Comprobamos que las proporciones se mantienen en los datasets de entrenamiento y test similares a las del dataset original



Preprocesado de datos. Buscamos variables con varianza cercana a cero (casi constantes) pues no aportan al clasificador con la función **nearZeroVar()**

****

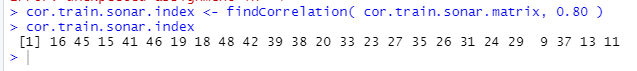
**# Eliminamos las columnas con varianza casi nula en caso de que haya alguna**

**# train.sonar.nz <- train.sonar[,-zero.var.train.sonar]**

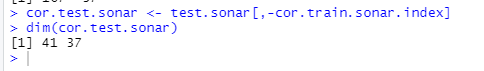
Ahora buscamos variables fuertemente correladas con la función **cor() y findCorrelation().**

****

**# Seleccionamos los indices de las columnas con una correlación mayor de 0,8**

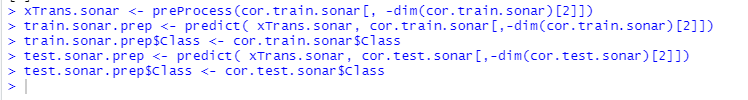
****

****

****

**El número de variables predictoras se reduce casi a la mitad quitando las que están correladas**

**Centramos y escalamos las variables para reducir la desviación con la función preProcess()**

****

**Generación de modelos**

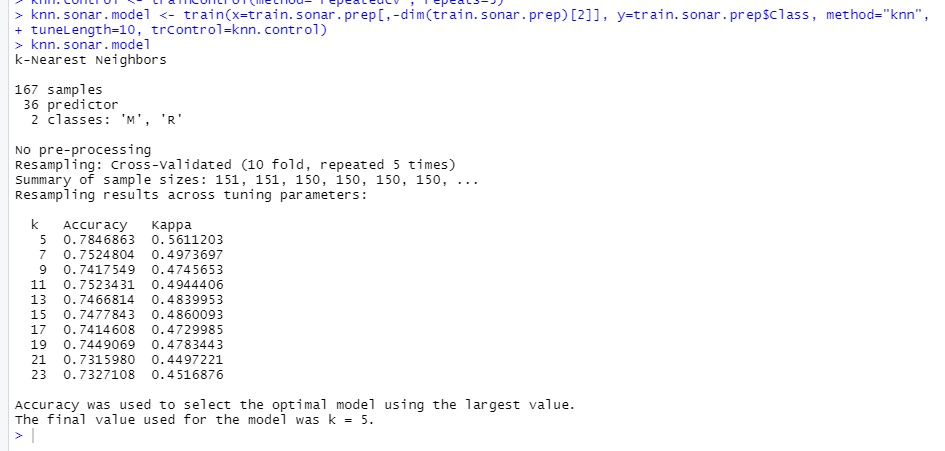
**1.Modelo KNN**

Entrenamos el modelo

# Aplicamos resampling de repeated k cross validation



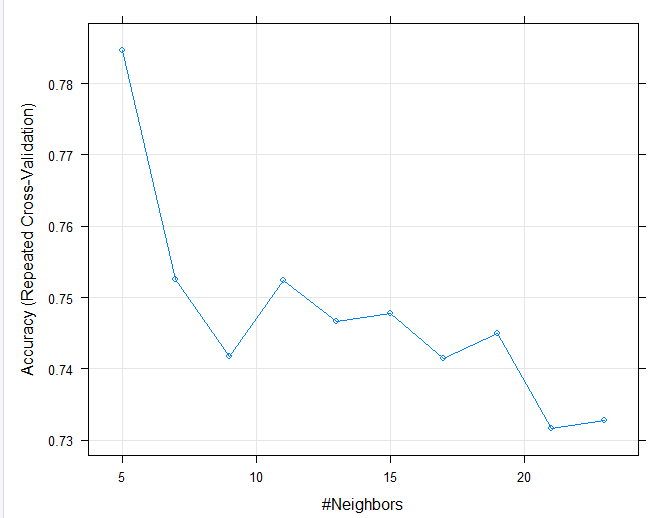
# Entrenamos el modelo



Caret realiza varias interacciones para determinar el valor óptimo de K en este caso y que no lo tengamos que hacer nosotros

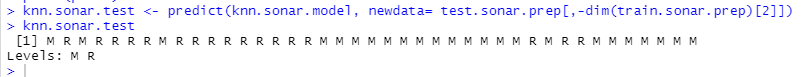
Visualizamos la salida del modelo KNN





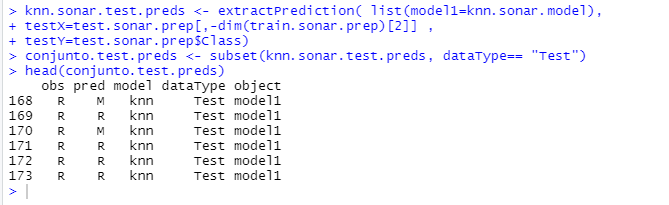
**Se confirma que el número óptimo de vecinos es k=5,** gracias a Caret hayar este número es muy sencillo

**Predicción Knn sobre el conjunto de test**

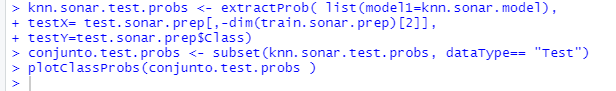
****

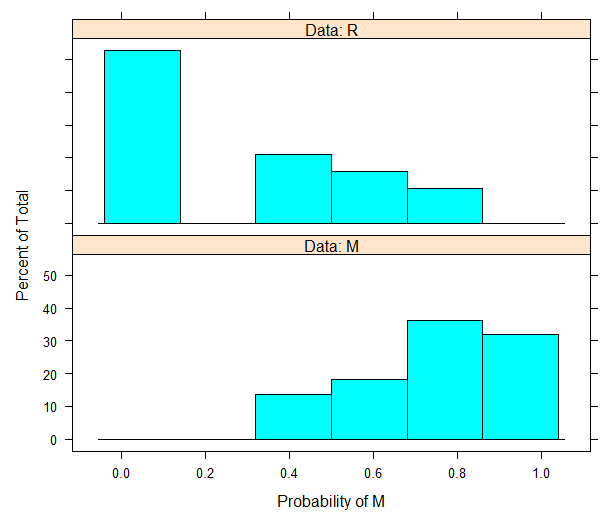
Comparación de predicciones, probabilidades sobre el conjunto de test

# Extracción de prediciones



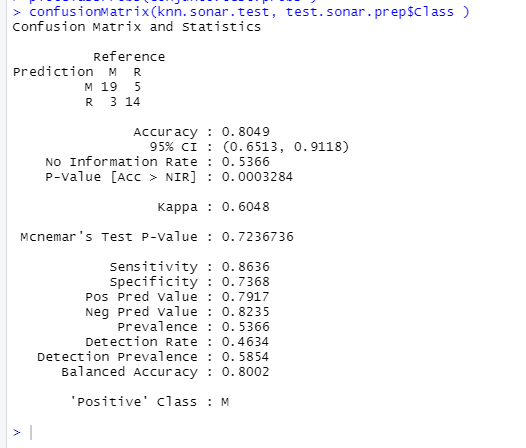
# Extraccion de probabilidades



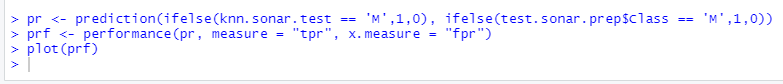


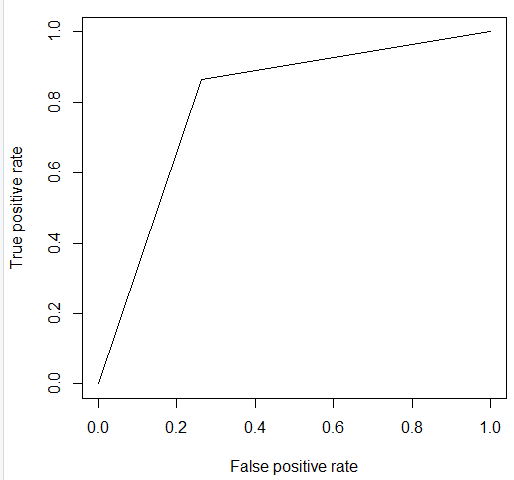
Vemos en la gráfica que las probabilidades de ser más certero prediciendo el valor M (1) son mayores

Evaluación del modelo, matriz de confusión y curvas ROC. Funciones **confusionMatrix()** y **prediction()**

****

**# Para poder pintar la curva ROC tenemos que pasar los valores del target a 0 y 1**

****

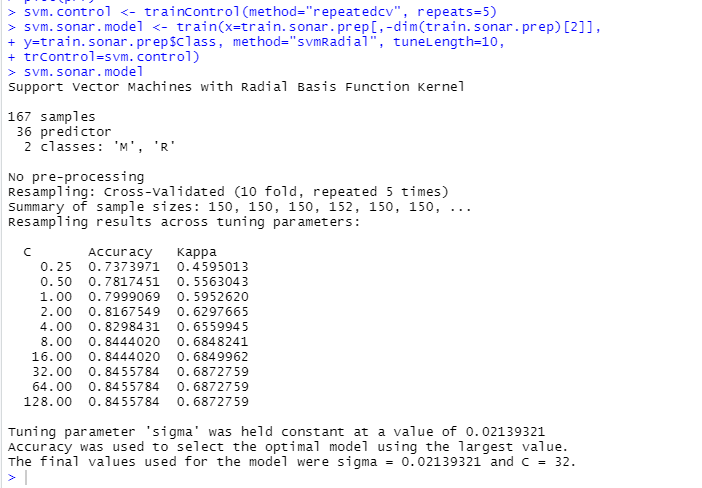
****

**2.Modelo SVM**

Repetimos los mismos pasos que para Knn

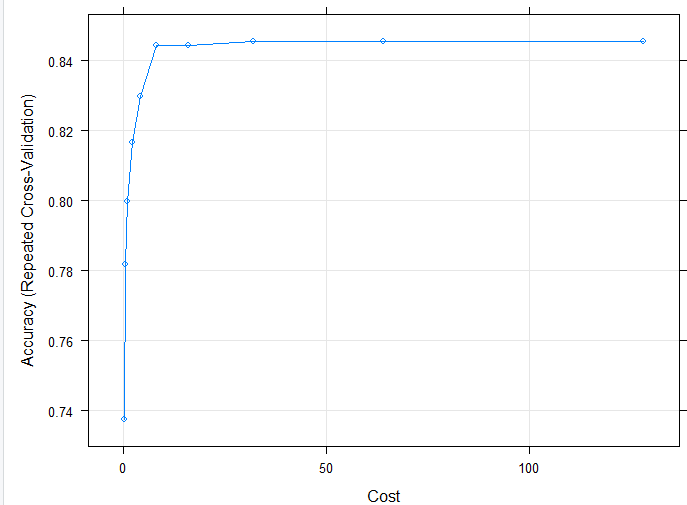
Utilizamos validación cruzada como técnica de remuestreo

Entrenamiento

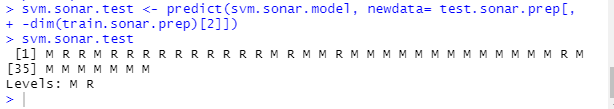


Visualización del modelo

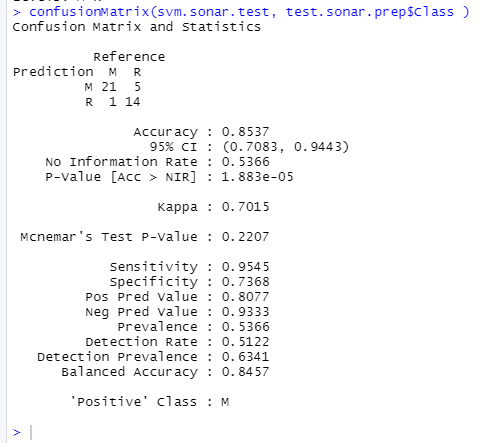


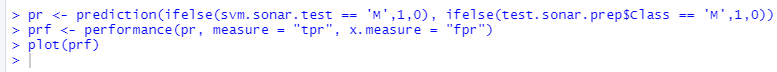


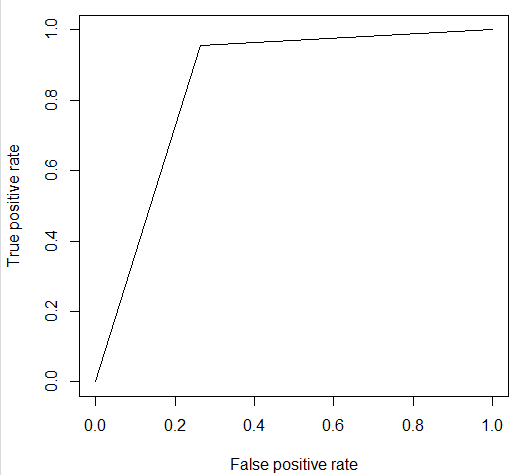
Predicción sobre nuevos valores



Evaluación del modelo, matriz de confusión y curvas ROC

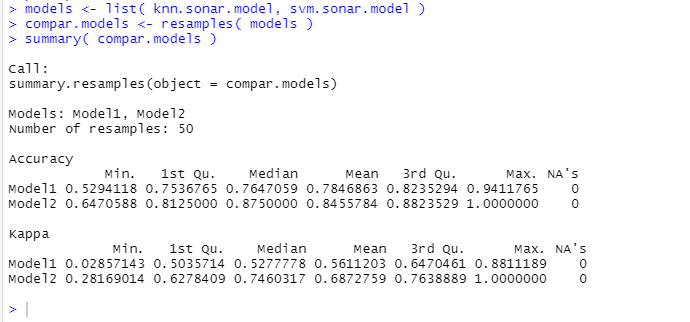






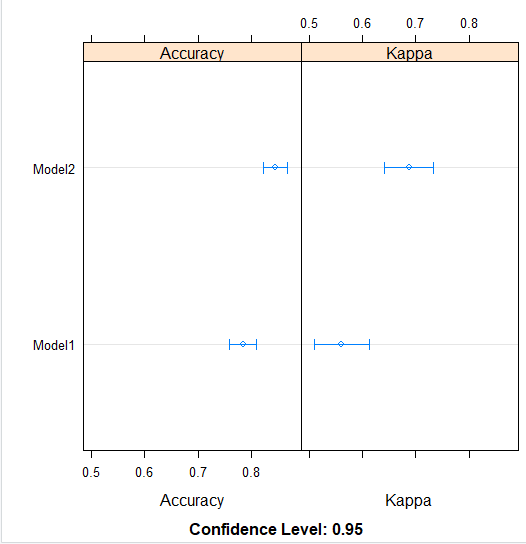
Comparación de Modelos: KNN vs SVM

**Model1=knn, Model2=SVM**

****

**Visualización del rendimiento de cada modelo. Model1=knn, Model2=SVM**

****

****

**Se ve claramente que Model2=SVM para este caso en concreto tiene mayor precisión.**

**Procesado de lenguaje natural con R**

Vamos a ver un completo ejemplo de cómo podemos usar R aplicado a un ejercicio de procesado de lenguaje natural (NLP en inglés), más concretamente en el campo de Topic Modeling o agrupación automática de palabras que se suele utilizar frecuentemente en la clasificación de documentos.

Para este ejercicio vamos a utilizar el paquete tm100 (Text Mining) que viene con un buen número de utilidades para tratamiento de texto y en varios idiomas.

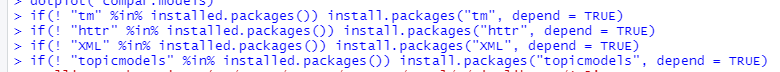
El objetivo del ejemplo es a partir de un listado de urls de páginas web extraer el título de las mismas y obtener grupos de palabras (topics) que las caractericen.

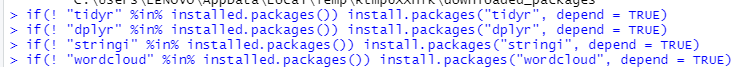
Para ello hemos construido un listado con las páginas web más vistas en España en las categorías de Música, Deporte y Viajes según similarweb.com101.

Os recomendamos también otros dos ejemplos similares con tweets de Twitter102 103 y uno más con artículos de Wikipedia104 muy interesantes para complementar el que veremos a continuación.

100

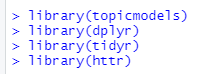
Cargamos las librerías necesarias para ejecutar el ejemplo de topic modeling con **tm: httr, XML, topicmodels, tidyr, dplyr, stringi, ggplot2 y wordcloud106**

****

****

****

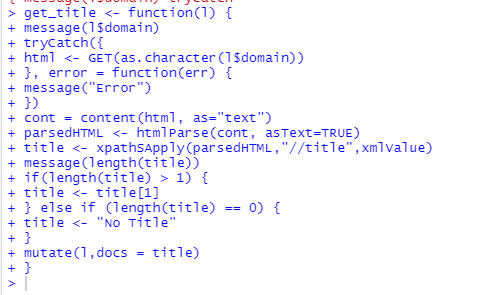
****

****

****

****

**Obtemos los títulos de cada una de las urls del listado referentes a Música, Deporte y Viajes**

****

**# El listado se encuentra en un fichero que hay que cargar**

**Pag 115**