# Facultad de ciencias

Departamento de Estadística

Carrera Ingeniería estadística


# Proyecto de título II

**Asignatura:** Proyecto de título II

**Título:** Minería de datos.


**Realizado por:** Cristian Friz.

Estudiante de Ingeniería Estadística

**Profesor Guía:** Dr. Francisco Novoa.

Departamento de Estadística

**Profesor Co-Guía:** Dr. Tarik Faouzi Nadim.

Departamento de Estadística


**Semestre:** Primer semestre de 2017

Departamento de Estadística

**Fecha:** Concepción, Septiembre 2017


UNIVERSIDAD DEL BÍO-BÍO

www.ubiobio.cl

# Métodos Predictivos Aplicados al Impacto de la Investigación Educativa Sobre la Práctica Docente.

Cristian Friz San Martin
Departamento de Estadística, Universidad del Bío-Bío

6 de Septiembre de 2017.

# Resumen

En el presente proyecto se realizó un análisis y aplicación de tres técnicas predictivas fundamentales en minería de datos: Árboles de Decisión, Redes Neuronales y Support Vector Machine (SVM).

En el estudio se consideró una muestra de 179 individuos, 62 docentes universitarios y 117 no universitarios de Granada (España), que fue el resultado de una publicación realizada por el profesor Tarik Faouzi quien gentilmente facilitó los datos de dicha investigación ("impacto de la investigación educativa") y propuso el tema del proyecto de título.

A la muestra de datos se les aplicaron las tres técnicas de minería de datos con el objetivo de elegir el mejor modelo.

El estudio muestra la potencialidad de la técnica de Support Vector Machine con respecto a las otras dos técnicas mencionadas anteriormente.

***Palabras claves***: árboles de decisión, redes neuronales, support vector machine, impacto de la investigación educativa.

# Agradecimientos

En primer lugar, quiero dar las gracias a mis padres, hermanas, abuelita y toda mi familia en general por el eterno apoyo que siempre he recibido de ellos. También quiero mencionar a mis compañeros por todos los momentos vividos. A mi novia por su incondicional ayuda. Por último, quiero agradecer a todos mis profesores en mi formación como ingeniero estadístico y en especial a mis tutores, al profesor Tarik Faouzi y Francisco Novoa por todo el soporte que me han brindado este tiempo y la paciencia que han tenido conmigo.

Gracias a todos.

# Índice

# Índice de figuras

# Índice de cuadros

# 1. Introducción

Es evidente que la educación no ha logrado posicionarse frente a otras disciplinas y áreas científicas, debido a esto se ha desarrollado un gran interés y preocupación en el mundo científico y por lo tanto han surgido muchas teorías acerca de la manera en que los docente desarrollan conocimientos errados sobre supuestos científicos inválidos y que terminan por enseñar prácticas educativas perniciosas para los estudiantes ($Karakus, Howard - Jones, Jay, 2015$).

Se sabe además que la Investigación en Educación realizó conocimientos solidos sobre las mejores prácticas en la década del siglo pasado, los docentes cuentan con algo más que sus preferencias para enseñar. En esta dirección , la investigación y la experiencia en los procesos de enseñanza y aprendizaje pueden explicar la evidencia que justifica los procesos de originalidad y mejora en el aula ($Fulllan - Hargreaves, 2014$).

Es por ello que este proyecto tiene como propósito desarrollar herramientas para medir el impacto de la investigación educativa sobre la práctica docente (IIE-PD).

En este proyecto se desarrollaron tres técnicas de predicción para medir el impacto: árboles de decisión, redes neuronales y support vector machine los cuales generaron modelos para calcular el impacto a través de los factores mencionados por Díaz-E; Fernández-Cano; T Faouzi y Henríquez Carlos.

# 2. Objetivos.

## 2.1. Objetivo general.

Crear un modelo estadístico basado en algunas herramientas de minería de datos que ayude a clasificar nueva información con el menor error posible.

## 2.2. Objetivos específicos.

1. Crear programas en el software R para las metodologías de Árbol de Decisión, Redes Neuronales y Support Vector Machine.
2. Analizar los resultados obtenidos de las tres técnicas de predicción.
3. Escoger el modelo que entregue mejores resultados.

# 3. Metodología.

## 3.1. Diseño.

El diseño de este estudio se caracteriza por ser *predictivo, cuantitativo*, tipo encuesta en donde se pretende estimar valores futuros o desconocidos de la variable de interés, que se denomina *variable objetivo o dependiente*, usando otras variables de la base de datos a las cuales llamaremos *variables independientes o predictivas.*

## 3.2. Descripción del instrumento de medida.

En el cuadro 1 se presentan las variables predictivas para explicar y analizar el **impacto de la investigación educativa sobre la práctica docente que es la variable dependiente que nos interesa en esta investigación**.

| ***Variables Independientes*** | ***Etiqueta*** |
|---|---|
| **TIPOLOGÍ** | 1= "colegio" , 2= "universidad". |
| **INSTITUC** | 1= "IES Padre Andrés Manjón", 2= "Ave María casa madre", 3= "Colegio Juan Ramón Jiménez", 4= "Colegio Fuente nueva", 5= "Instituto Padre Suárez", 6= "Virgen de las Nieves", 7= "Cristo de la Yedra", 8= "Santa Cristina", 9= "Miguel Hernández", 10= "Juan XXIII", 11= "Universidad de Granada". |
| **SEXO** | 1="Mujer", 2="Hombre". |
| **edad** | 1="menor que 30", 2="de 30 a 40", 3="41 a 50", 4="51 a 60", 5="mayor de 60". |
| **expdocente** | **Años de experiencia docente**. 1="menos de 7 años", 2="8 y 14 años", 3="15 y 24 años", 4="25 y 30 años", 5="más de 31 años". |
| **nivelactual** | **Nivel educativo Actual**. 1="un sólo nivel", 2="más de un nivel". |
| **EXPINV** | **Experiencia en Investigación Educativa**. 1="Ninguna"2="Colaboración ocasional", 3="Colaboración continua sin equipo", 4="Miembro de equipo de investigación", 5="Coordinador de proyectos", 6="Más de una alternativa". |
| **FORINV** | **Formación previa en Investigación Educativa**. 1="Ninguna", 2="Cursos y seminarios extra universitarios", 3="Formación Universitaria", 4="Formación de posgrado", 5="Título de doctor", 6="Más de una alternativa". |

Cuadro 1: Variables Independientes.

Como primera versión se diseñó escala criterial de cinco puntos para la variable dependiente (IIE-PD), a valorar de 1 a 5 (1=Muy en desacuerdo, 2=En desacuerdo, 3=Indiferente, 4=De acuerdo y 5=Muy de acuerdo) y fue administrada individualmente como un cuestionario a docentes universitarios y no universitarios (Díaz, 2010).

# 4. Hipótesis del estudio.

La finalidad de este proyecto de título es aplicar herramientas estadísticas que nos permitan confirmar o rechazar la siguiente hipótesis de estudio:*el impacto de la investigación influye sobre la docencia.*

# 5. Muestra.

Para este proyecto, se recurrió a datos de la muestra utilizada por Díaz, Fernández- Cano, T Faouzi y Henríquez (2015), realizando un análisis secundario de los mismos. El muestreo fue no probabilístico y se obtuvo una muestra compuesta por 179 encuestados cuantificada por 62 docentes universitarios y 117 no universitarios.

# 6. Técnicas de análisis de datos.

A partir del análisis de datos generado por la muestra disponible a la que se le aplicó la escala del impacto de la investigación educativa sobre la práctica docente (EI/IE-PD), se continuó con el análisis de árboles de decisión, redes neuronales y support vector machine con la finalidad de generar un modelo predictivo que se ajuste bien a los datos y que permita explicar la variable de interés (ítem 43), en función de la variable latente mencionada previamente. En este análisis se utilizó el software libre del programa R-Proyect, versión 3.2.5.

# 7. Árboles de decisión.

## 7.1. Aplicando Árboles de decisión.

Un árbol de decisión es un conjunto de condiciones organizadas en una estructura jerárquica, de tal manera que la decisión final a tomar se puede determinar siguiendo las condiciones que se cumplen desde la raíz del árbol hasta alguna de sus hojas. Además este método de aprendizaje es quizás uno de los más fáciles de utilizar y entender ya que la representación gráfica del árbol facilita mucho la comprensión del modelo [Orallo et al., 2004].

De acuerdo al tipo de variable dependiente existen dos tipos de árboles de decisión: Árbol de clasificación y de regresión. El árbol de decisión por clasificación se usa en procesos binarios de categorías tanto para variables nominales u ordinales. Mientras que el árbol de regresión se aplica cuando la variable dependiente es de tipo continuo.

Por lo tanto debido a que la variable dependiente de los datos es de tipo categóricos, entonces emplearemos solo árboles de decisión por clasificación.

El algoritmo o sistema de aprendizaje de árboles de decisión CART (*Classification and Regression Trees*) son métodos "divide y vencerás" que construyen árboles binarios y se basan en el criterio de partición GINI (el cual se trató en proyecto de título I) y que sirve tanto para clasificación como para regresión. La poda se basa en una estimación de complejidad del error (error-complexity). Generalmente CART se puede encontrar en paquetes de minería de datos.

## 7.2. Poda de árboles.

Cuando se construye un árbol de decisión, muchas de las ramas reflejarán anomalías en los datos de entrenamiento debido al ruido o valores atípicos. Los métodos de poda de árboles abordan este problema de sobreajuste. Tales métodos suelen utilizar medidas estadísticas para eliminar las ramas menos fiables. Un árbol no podado y una versión podada de él se muestran en la Figura 1 que se indica más adelante. Los árboles ramificados tienden a ser más pequeños y menos complejos y, por lo tanto, más fácil de comprender. Por lo general, son más rápidos y mejores al clasificar correctamente los datos de pruebas independientes (es decir, de las tuplas previamente invisibles) que los árboles sin podar.

¿Cómo funciona la poda de árboles? Hay dos enfoques comunes para la poda de árboles: prepoda y pospoda.

En el enfoque de prepoda, un árbol es podado deteniendo su construcción temprano (por ejemplo, decidiendo no dividir o dividir el subconjunto de tuplas de entrenamiento en un nodo dado).

El segundo enfoque más común es la pospoda, que elimina los subárboles de un árbol completamente crecido. Un subárbol en un nodo dado se poda quitando sus ramas y reemplazándolas con una hoja. La hoja se etiqueta con la clase más frecuente entre el subárbol que se está reemplazando. Por ejemplo, observe el subárbol en el nodo A3?, en el árbol no podado de la Figura 1. Supongamos que la clase más común dentro de este subárbol es clase B. En la versión podada del árbol, el subárbol en cuestión es podado reemplazándolo con la hoja clase B [Han et al., 2011].

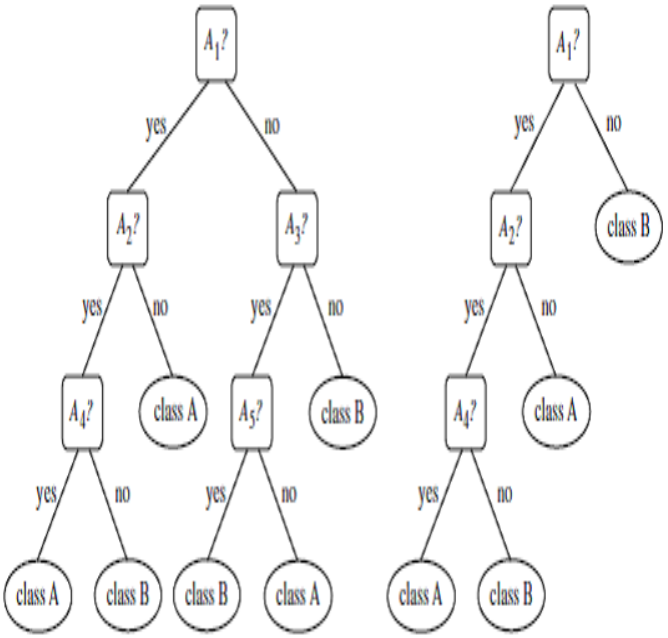


Figura 1: Un árbol de decisiones no podado y podado.

## 7.3. Pasos a seguir para la construcción de un árbol de decisión.

1. **Disposición de los datos:** se crean datos de entrenamiento y de validación por medio de una muestra aleatoria. Los datos de entrenamiento sirven para la creación del modelo y la estimación de los parámetros y los de validación para realizar pruebas de verificación del error de predicción del modelo basado en la técnica de validación cruzada.

2. **Criterio de corte:** existe una variedad de criterios de corte como por ejemplo Entropía, DKM, Error esperado y GINI. Como se trabaja con el algoritmo CART se utiliza el índice GINI. R emplea este corte por defecto.

3. **Criterio de poda:** R entrega valores en una tabla a un parámetro de complejidad, se debe elegir el que tenga un menor xerror de la tabla, ver por ejemplo cuadro 2.

4. **Evaluación del modelo:** una vez que se ha podado el árbol hay que validar los resultados con los datos de validación en donde se mide el poder de clasificación del modelo. La matriz de confusión mide los porcentajes bien clasificados por ende se utilizará dicha matriz.

5. **Resultados:** una vez validados los porcentajes de clasificación, se complementa con un promedio de la estimación del error, por medio de la técnica de validación, en este caso se utilizara la Validación Cruzada con los datos del conjunto de validación. Esta técnica entrega un error medio al repetir los ensayos una cierta cantidad de veces.

## 7.4. Resultados arboles de decisión.

### 7.4.1. Introducción.

Se procede a construir el modelo para predecir a través de ciertas variables. En este capítulo se desarrollaran los pasos mencionados en el capítulo anterior.

El objetivo general de este proyecto es crear un modelo estadístico que permita predecir a través de ciertas variables la probabilidad de estar en desacuerdo o de acuerdo si el impacto de la investigación educativa influye sobre la docencia, para ello se trabajó con el software estadístico R, los comandos utilizados están en el anexo.

### 7.4.2. Creación del modelo (árbol) sin podar.

La siguiente salida muestra el modelo teórico del árbol de decisión mediante la función rpart con los datos de entrenamiento y el método de clasificación.

```
n= 161

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 161 58 2 (0.3602484 0.6397516)
   2) INSTITUC=1,3,5,6,7,9,11 99 47 2 (0.4747475 0.5252525)
     4) expdocente=1,2,5 69 29 1 (0.5797101 0.4202899)
       8) edad=1,2,3,4 60 22 1 (0.6333333 0.3666667)
        16) FORINV=1,6 40 12 1 (0.7000000 0.3000000) *
        17) FORINV=2,3 20 10 1 (0.5000000 0.5000000)
          34) INSTITUC=1,3,5,6,9 12  4 1 (0.6666667 0.3333333) *
          35) INSTITUC=7 8  2 2 (0.2500000 0.7500000) *
       9) edad=5 9  2 2 (0.2222222 0.7777778) *
     5) expdocente=3,4 30  7 2 (0.2333333 0.7666667) *
   3) INSTITUC=2,8,10 62 11 2 (0.1774194 0.8225806) *
```

Se indica n como el número de observaciones que alcanzan a cada nodo, loss es el número de observaciones que se clasifican mal, yval es el valor de clasificación que se toma como referencia (“1”o “2”, en este caso) e yprob las probabilidades de ambas clases (el primer valor se refiere a la probabilidad de alcanzar el valor “1” y el segundo a la de alcanzar el valor “2”).

El árbol es leído desde el nodo raíz que es marcado con el número 1 y provee alguna información de los datos en este nodo. Se puede observar que hay 161 muestras para este nodo (corresponde a los datos de entrenamiento utilizados para generar el árbol), cada nodo del árbol contiene dos ramas, las cuales están relacionadas con el resultado de la prueba sobre la variable predictora. Por ejemplo, desde el nodo raíz se tiene una rama o la primera decisión (etiquetada por R con “2)”) para los casos donde INSTITUC= $1, 3, 5, 6, 7, 9, 11$, se clasifica como 2, a este nodo llegan 99 observaciones de las cuales 47 están mal clasificadas y la probabilidad de que yval(2) sea 1 es de 0.4747475 y de que sea 2 es de 0.5252525. Notar que el nodo 3 es la otra rama del árbol.

Por otro lado la salida:

```
1) root 161 58 2 (0.3602484 0.6397516)
```

Significa que para el nodo raíz tenemos que del total de 161 observaciones un 36 % está en desacuerdo y el 64 % está de acuerdo.

Para la salida:

```
4) expdocente=1,2,5 69 29 1 (0.5797101 0.4202899)
```

Significa que para el nodo 4 si expdocente= $1, 2, 5$ se clasifica como 1, hay 69 datos, que se han perdido o clasificado mal 29 en esa división, además la probabilidad de que yval (1) sea 1 es de 0.5797101 y que sea 2 es 0.4202899, es decir, de un total de 69 datos un 58 % está en desacuerdo y un 42 % está de acuerdo.

También nos dice que para el nodo 5 si expdocente= $3, 4$ se clasifica como 2, hay 30 datos, se han perdido o clasificado mal 7 en esa división, además la probabilidad de que yval (2) sea 1 es 0.2333333 y que sea 2 es 0.7666667, es decir, de un total de 30 datos un 23 % está en desacuerdo y un 77 % está de acuerdo.

La salida:

```
 8) edad=1,2,3,4 60 22 1 (0.6333333 0.3666667)
     9) edad=5 9  2 2 (0.2222222 0.7777778) *
```

Nos dice que para el nodo 8 tenemos que si edad= $1, 2, 3, 4$ se clasifica como 1, hay 60 datos, se han perdido 22 en esa división, además la probabilidad de que yval (1) sea 1 es 0.6333333 y que sea 2 es 0.3666667, es decir, de un total de 60 datos un 63 % está en desacuerdo y un 37 % está de acuerdo.

De igual forma se tiene que para el nodo 9 tenemos que si edad= 5 se clasifica como 2, hay 9 datos, se han perdido 2 en esa división, además la probabilidad de que yval (1) sea 1 es 0.2222222 y que sea 2 es 0.7777778, es decir, de un total de 9 datos un 22 % está en desacuerdo y un 78 % está de acuerdo.

```
16) FORINV=1,6 40 12 1 (0.7000000 0.3000000) *
       17) FORINV=2,3 20 10 1 (0.5000000 0.5000000)
```

Nos dice que para el nodo 16 tenemos que si FORINV= $1, 6$ se clasifica como 1, hay 40 datos, se ha perdido 12 en esa división, además la probabilidad de que yval (1) sea 1 es 0.7000000 y que sea 2 es 0.3000000, es decir, de un total de 40 datos un 70 % está en desacuerdo y un 30 % está de acuerdo.

Para el nodo 17 tenemos que si FORINV= $2, 3$ se clasifica como 1, hay 20 datos, se han perdido 10 en esa división, además la probabilidad de que yval (1) sea 1 es 0.5000000 y que sea 2 es 0.50000000, es decir, de un total de 20 datos un 50 % está en desacuerdo y un 50 % está de acuerdo.

```
34) INSTITUC=1,3,5,6,9 12  4 1 (0.6666667 0.3333333) *
         35) INSTITUC=7 8  2 2 (0.2500000 0.7500000) *
```

Indica que Para el nodo 34 si INSTITUC= $1, 3, 5, 6, 9$ se clasifica como 1, hay 12 datos, se han perdido 4 en esa división, además la probabilidad de que yval(1) sea 1 es 0.6666667 y que sea 2 es 0.3333333, es decir, de un total de 12 datos un 67 % está en desacuerdo y un 33 % está de acuerdo.

Para el nodo 35 si INSTITUC= 7 se clasifica como 2, hay 8 datos, se han perdido 2 en esa división, además la probabilidad de que yval(2) sea 1 es 0.2500000 y que sea 2 es 0.7500000, es decir, de un total de 8 datos un 25 % está en desacuerdo y un 75 % está de acuerdo.

Estas pruebas continúan hasta que un nodo hoja es alcanzado. Estos nodos son marcados con un asterisco por R. Estas hojas tienen las predicciones del árbol.

El valor encontrado para la variable objetivo hasta la hoja que se ha alcanzado es la predicción del árbol. También se puede obtener una representación gráfica del árbol, presentado en figura 2.

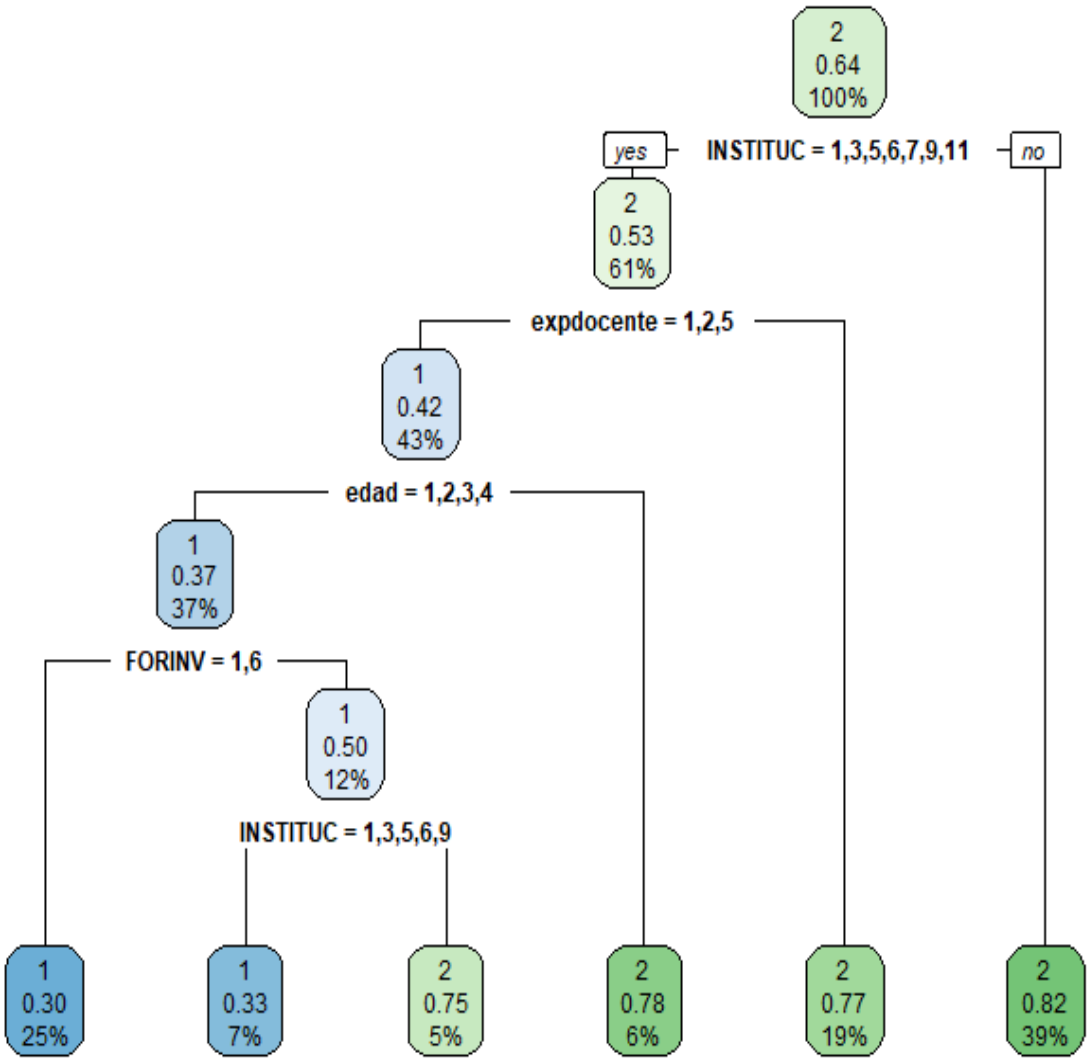


Figura 2: Modelo árbol sin podar.

### 7.4.3. Validación cruzada como técnica para la evaluación del modelo de árbol de decisión.

Los métodos de aprendizaje permiten construir modelos o hipótesis a partir de un conjunto de datos (evidencias). Generalmente es necesario evaluar la calidad de las hipótesis que sean lo más exacto posible. Por ejemplo, si en la aplicación de un modelo hay un error en la predicción, acarrea importantes consecuencias (por ejemplo, en la detección de células cancerígenas), es de suma importancia conocer con exactitud el nivel de precisión del modelo aprendido.

Ahora si se tiene un modelo predictivo y se quiere saber qué tan buena será la predicción con datos futuros, una forma de lograrlo es usando la técnica de validación cruzada (Cross validation) con un cierto número de iteraciones, por ejemplo con $k$ iteraciones.

Este método consiste en dividir el conjunto de la evidencia en k subconjuntos disjuntos de similar tamaño. Entonces, se aprende la hipótesis utilizando el conjunto formado por la unión de $k-1$ subconjuntos y el subconjunto restante se emplea para calcular un error de muestra parcial. Este procedimiento se repite $k$ veces, utilizando siempre un subconjunto diferente para estimar el error de muestra parcial. El error de muestra final se calcula como la media aritmética de los $k$ errores de muestras parciales. De esta manera el resultado final recoge la media de los experimentos con $k$ subconjuntos de pruebas independientes [Orallo et al., 2004].

En resumen esta técnica consiste en dividir los datos en varios set de datos y luego elegir uno de los set para testear y el resto para entrenar. Esto se hace de forma repetida hasta testear cada set de datos, guardando el resultado de cada iteración en una tabla para luego analizar la eficiencia de la predicción.

En el cuadro 2 se muestra una tabla de validación cruzada, la cual indica:

1. Qué variables fueron útiles para crear el árbol.
2. Da el Root node error, que podemos interpretar como varianza total.
3. CP: cost-complexity factor.
   Utilizaremos los parámetros de complejidad (CP) como una penalización para controlar el tamaño del árbol. En resumen, cuanto mayor es el parámetro de complejidad, menos decisiones contiene el árbol (nsplit).
4. nsplit: número de divisiones (tamaño del árbol).
5. rel error: residuo o error relativo.
6. xerror y xstd: error promedio y error estándar del error relativo, respectivamente, calculados por validación cruzada. Fijarse que, aunque rel error siempre disminuye al aumentar nsplit, xerror deja de hacerlo relativamente pronto (sobreajuste). Una estrategia típica es hacer crecer mucho el árbol y luego podarlo para evitar sobreajustes (sobreparametrización).

```
Classification tree:
rpart(formula = p43 ~ ., data = train, method = "class")

Variables actually used in tree construction:
[1] edad       expdocente FORINV     INSTITUC

Root node error: 58/161 = 0.36025

n= 161
```

| Nro del Árbol | CP | nsplit | rel error | xerror | xstd |
|---|---|---|---|---|---|
| 1 | 0.094828 | 0 | 1.00000 | 1.00000 | 0.10502 |
| 2 | 0.086207 | 2 | 0.81034 | 1.10345 | 0.10706 |
| 3 | 0.034483 | 3 | 0.72414 | **0.91379** | 0.10280 |
| 4 | 0.010000 | 5 | 0.65517 | 0.94828 | 0.10375 |

Cuadro 2: Elección del CP.

Una regla para seleccionar el mejor árbol consiste en mirar el error estimado para la validación cruzada (columna "xerror") y su desviación estándar (columna "xstd").

Así, observamos el menor xerror que es 0.91379 con nsplit=3, lo que me indica que el árbol debe podarse a ese nivel con un CP=0.034483.

Por lo tanto el árbol producido por la función rpart() es el tercer árbol del cuadro 2. Este árbol incluye 3 pruebas y tiene un error relativo de 0.72414.

La información sobre el parámetro CP se puede visualizar en LA Figura 3.

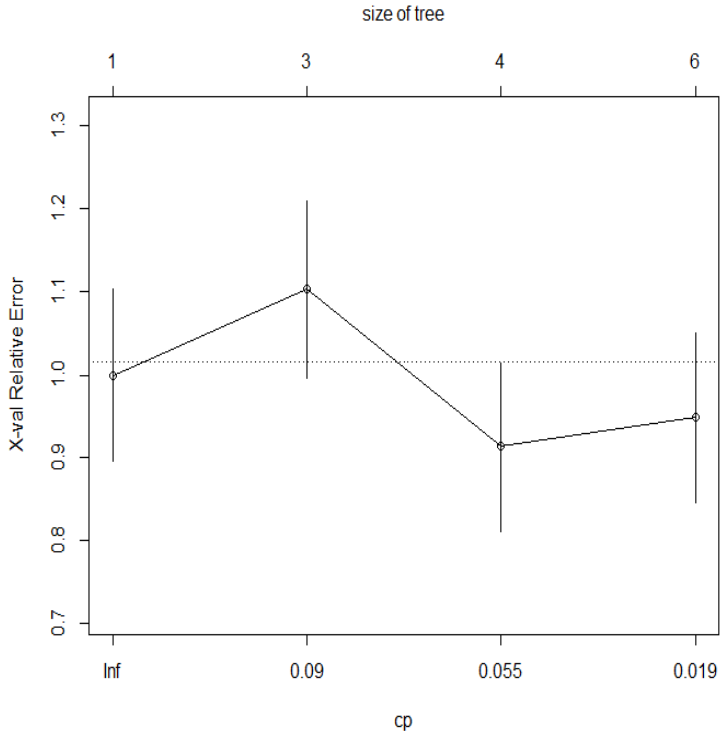


Figura 3: Gráfico parámetro de complejidad CP.

El eje horizontal inferior representa el CP, el eje vertical los valores de xerror y el eje horizontal superior es el tamaño del árbol.

A continuación se obtiene el árbol de decisión de clasificación teórico podado; es decir, un resumen construido con la función prune de rpart() y las interpretaciones de los nodos.

```
n= 161

node), split, n, loss, yval, (yprob)
      * denotes terminal node

1) root 161 58 2 (0.3602484 0.6397516)
  2) INSTITUC=1,3,5,6,7,9,11 99 47 2 (0.4747475 0.5252525)
    4) expdocente=1,2,5 69 29 1 (0.5797101 0.4202899)
      8) edad=1,2,3,4 60 22 1 (0.6333333 0.3666667) *
      9) edad=5 9  2 2 (0.2222222 0.7777778) *
    5) expdocente=3,4 30  7 2 (0.2333333 0.7666667) *
  3) INSTITUC=2,8,10 62 11 2 (0.1774194 0.8225806) *
```

La salida:

```
1) root 161 58 2 (0.3602484 0.6397516)
```

Significa que para el nodo raíz tenemos que del total de 161 observaciones un 36 % está en desacuerdo y el 64 % está de acuerdo.

La salida:

```
2) INSTITUC=1,3,5,6,7,9,11 99 47 2 (0.4747475 0.5252525)
     3) INSTITUC=2,8,10 62 11 2 (0.1774194 0.8225806) *
```

Nos dice que para el nodo 2 tenemos que si INSTITUC= $1, 3, 5, 6, 7, 9, 11$ se clasifica como 2, hay 99 datos, se han perdido 47 en esa división, además la probabilidad de que yval (2) sea 1 es 0.4747475 y que sea 2 es 0.5252525, es decir, de un total de 47 datos un 47 % está en desacuerdo y un 53 % está de acuerdo.

Para el nodo 3 tenemos que si INSTITUC= $2, 8, 10$ se clasifica como 2, hay 63 datos, se han perdido 11 en esa división, además la probabilidad de que yval (1) sea 1 es 0.1774194 y que sea 2 es 0.8225806, es decir, de un total de 62 datos un 18 % está en desacuerdo y un 82 % está de acuerdo.

La salida:

```
  4) expdocente=1,2,5 69 29 1 (0.5797101 0.4202899)
  5) expdocente=3,4 30  7 2 (0.2333333 0.7666667) *
```

Nos dice que en el nodo 4 tenemos que si expdocente= $1, 2, 5$ se clasifica como 1, hay 69 datos, se han perdido 29 en esa división, además la probabilidad de que yval (1) sea 1 es 0.5797101 y que sea 2 es 0.4202899, es decir, de un total de 69 datos un 58 % está en desacuerdo y un 42 % está de acuerdo. .

El nodo 5 es la otra rama del árbol y nos dice que si expdocente= $3, 4$ se clasifica como 2, hay 30 datos, se han perdido 7 en esa división, ademas además la probabilidad de que yval (2) sea 1 es 0.2333333 y que sea 2 es 0.7666667, es decir, de un total de 30 datos un 23 % está en desacuerdo

y un 77 % está de acuerdo.

Finalmente la salida:

```
8) edad=1,2,3,4 60 22 1 (0.6333333 0.3666667) *
     9) edad=5 9  2 2 (0.2222222 0.7777778) *
```

Significa que para el nodo 8 tenemos que si edad= $1, 2, 3, 4$ se clasifica como 1, hay 60 datos, se han perdido 22 en esa división, además la probabilidad de que yval (1) sea 1 es 0.6333333 y que sea 2 es 0.3666667, es decir, de un total de 60 datos un 63 % está en desacuerdo y un 37 % está de acuerdo.

Para el nodo 9 tenemos que si edad= 5 se clasifica como 2, hay 9 datos, se han perdido 2 en esa división, además la probabilidad de que yval (2) sea 1 es 0.2222222 y que sea 2 es 0.7777778, es decir, de un total de 9 datos un 22 % está en desacuerdo y un 78 % está de acuerdo.

La figura 4 muestra el árbol de decisión podado. El árbol enseña los nodos, con el nombre de la clasificación que tiene más probabilidades en ese nodo.

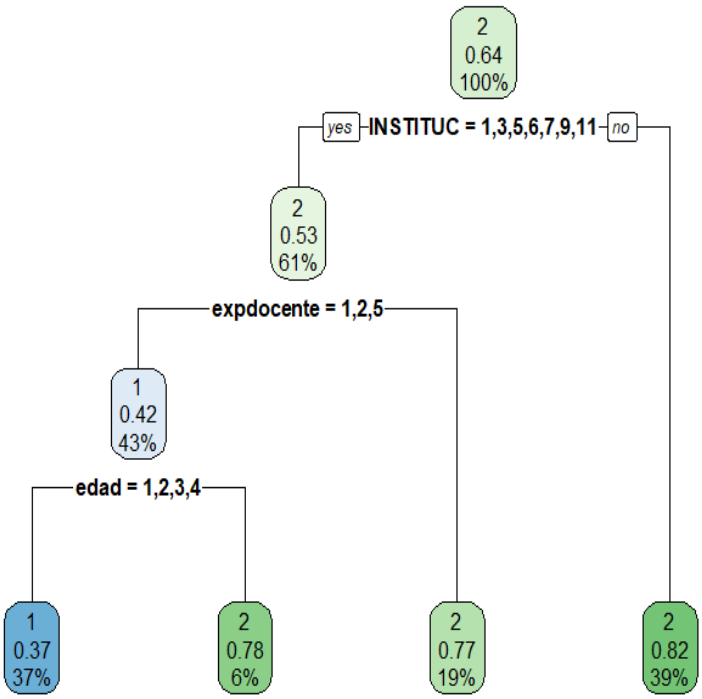


Figura 4: Modelo árbol podado (óptimo).

### 7.4.4. Evaluación del modelo.

Una vez podado el árbol de decisión se evalúa la predicción del modelo obtenido. Para ello utilizamos la matriz de confusión explicada en el capítulo 8.

A continuación, sobre el conjunto de validación, test, se muestran los resultados de la matriz de confusión para el modelo de árboles de decisión.

| | | **Real** | | |
|---|---|---|---|---|
| | | **de acuerdo** | **en desacuerdo** | **Total** |
| **Estimado** | **de acuerdo** | 1 | 2 | 3 |
| | **en desacuerdo** | 4 | 11 | 15 |
| | **Total** | 5 | 13 | 18 |

Cuadro 3: Resultados de la matriz de confusión para modelo de árboles de decisión.

La matriz de confusión nos indica que se estimaron 41 observaciones clasificadas como de acuerdo con que el impacto de la investigación educativa influye sobre la docencia de las cuales 29 estarían bien clasificadas y 12 mal clasificadas. Además señala que 38 observaciones se estimaron como en desacuerdo con que el impacto de la investigación educativa influye sobre la docencia de las cuales 19 estarían bien clasificadas y 19 mal clasificadas.

Así concluimos que la ***precisión***; es decir, observaciones bien clasificadas del modelo en la matriz de clasificación corresponde a un 66.67 %, por lo tanto, el **error de predicción es de un 33.33 %**

Para validar los resultados y verificar el error de predicción se utiliza la técnica de validación cruzada, también explicada en la sección 7.4 y que también nos servirá para validar Redes Neuronales y Support Vector Machine.

A continuación en la figura 5 se muestra la realización de 10 iteraciones y el porcentaje de predicción en cada iteración, obteniéndose un promedio de la eficiencia general del modelo predictivo de árboles de decisión.

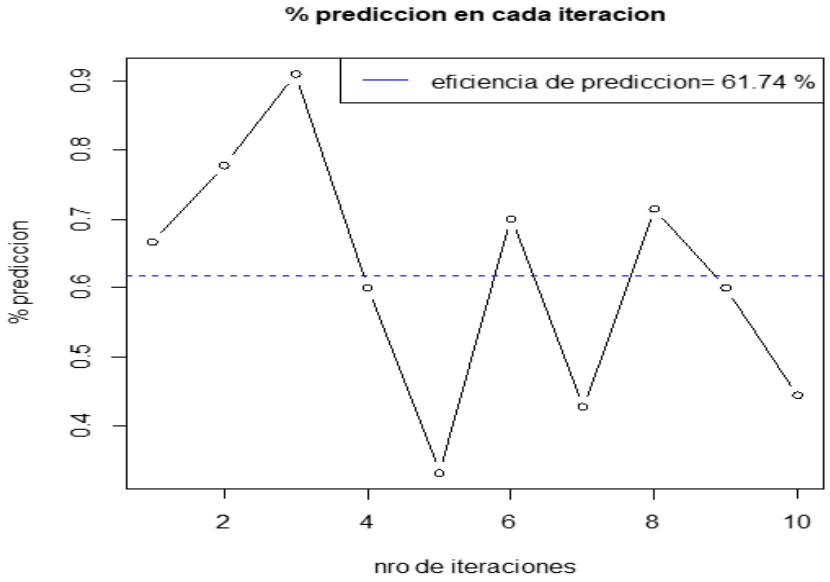


Figura 5: Promedio del error de predicción .

Por lo tanto se obtuvo un ***promedio del error de predicción de un 38.26 %*** **aproximadamente.**

# 8. Redes neuronales.

## 8.1. Introducción

Las redes neuronales son modelos computacionales o métodos de aprendizaje cuya finalidad es la de emular procesadores biológicos de la información y que surgieron como intento de conseguir formalizaciones matemáticas acerca de la estructura y el comportamiento del cerebro humano. [Dreyfus, 2005]

Aunque para resolver algunos problemas existen métodos estadísticos que ofrecen muy buenos resultados, también se pueden aplicar modelos de redes neuronales artificiales (RNA), los cuales son sistemas de conexión que se encuentran dentro del campo de la Inteligencia Artificial y que dependiendo de su arquitectura neuronal se pueden obtener diferentes aplicaciones y utilizarse para, la compresión de información , la reducción de la dimensionaldad, el agrupamiento, la clasificación, la visualización, etc.

Estos modelos pueden suponer una reducción en el tiempo de cálculo, no tener que hacer suposiciones sobre las distribuciones de los datos, no tener que determinar expresiones matemáticas que definan el problema a resolver o, simplemente, ser un método de solución alternativo a las técnicas estadísticas disponibles.

En la década de los 90 surgieron una gran cantidad de trabajos teóricos que comparaban las RNA con las estadísticas encontrándose similitudes entre ambas, Debido a esto se presentan los cuadros 4 y 5 para mostrar las diferentes terminologías para un mismo objeto. [Montaño Moreno et al., 2002]

| Terminología estadística | Terminología de redes neuronales |
|---|---|
| Observación | Patrón |
| Muestra | Datos de entrenamiento |
| Muestra de validación | Datos de validación, test |
| Variables explicativas | Variables de entrada |
| Variable de respuesta | Variable de salida |
| Modelo | Arquitectura |
| Residual | Error |
| Error aleatorio | Ruido |
| Estimación | Entrenamiento, aprendizaje |
| Interpolación | Generalización |
| Interacción | Conexión funcional |
| Coeficientes | Pesos de conexión |
| Constante | Peso umbral |
| Regresión y análisis discriminante | Aprendizaje supervisado o heteroasociación |
| Reducción de datos | Aprendizaje no supervisado o autoasociación |
| Análisis de clúster | Aprendizaje competitivo |

Cuadro 4: Equivalencia en la terminología estadística y de redes neuronales.

| Modelo estadístico | Modelo de red neuronal |
|---|---|
| Regresión lineal múltiple | Perceptrón simple con función lineal |
| Regresión logística | Perceptrón simple con función logística |
| Función discriminante lineal | Perceptrón simple con función umbral |
| Regresión no lineal múltiple | Perceptrón multicapa con función lineal en la salida |
| Función discriminante no lineal | Perceptrón multicapa con función logística enla salida |
| Análisis de componentes principales | Regla de Oja , Perceptrón multicapa autoasociativo |
| Análisis de clusters | Mapas autoorganizados de Kohonen |
| K vecinos más cercanos | Learning Vector Quantization (LVQ) |
| Regresión kernel | Funciones de Base Radial(RBF) |

Cuadro 5: Equivalencia entre modelos estadísticos y modelos de red neuronal.

Existen dos tipos de RNA,las que emplean aprendizaje supervisado (perceptrón multicapa y redes de Función de Base Radial) y las que no emplean aprendizaje supervisado (regla de Hebb y aprendizaje competitivo), ambas descritas en Proyecto De Título I.

### 8.1.1. Resumen de los resultados principales.

Se utilizará la modelización matemática basada en las Redes Neuronales Artificiales.

Con el programa R se modelara una red neuronal sobre la base de datos que expliquen la variable respuesta p43 (el impacto de la investigación educativa sobre la docencia).

El paquete **neuralnet** permite hacer configuraciones flexibles a través de la opción del error y la función de activación. Por otra parte, también se puede llevar a cabo el cálculo de pesos generalizados.

Además, se utilizaron diferentes funciones tales como:

1. plot.nn: para el trazado de la red neuronal.

2. gwplot: para el trazado de los pesos generalizados.
3. compute: para el cálculo de la red ya entrenada.
4. confidence.interval: para el cálculo de un intervalo de confianza para los pesos.
5. prediction: para el cálculo de las predicciones.

La función **compute**, calcula las salidas de todas las neuronas para los vectores de covarianza arbitrarias específicas dadas una red neuronal entrenada.

Antes de pasar la red neuronal por la fase de aprendizaje, se creara la red neuronal (Neural Network) para luego ser entrenada.

Para crear la red neuronal se utiliza la función o paquete **neuralnet** para así crear un buen modelo. Lo que se quiere es crear un modelo que describa el conjunto de datos en función de las variables independientes, es decir, que en función de las características de cada individuo se nos diga si está o no se está de acuerdo en el impacto de la investigación educativa sobre la práctica docente (la variable respuesta es binaria).

La muestra se divide en 3 partes cuando se tienen muchos datos (600, 1000, 2000,3000,..., etc.) para así evitar el problema de sobreajuste. Así con una muestra podemos hacer la fase de aprendizaje del modelo y con otra la fase de ejecución.

### 8.1.2. Creación de los modelos y resultados.

Se explicara la variable dependiente p43 a partir de las variables independientes TIPOLOGI, INSTITUC, SEXO, edad, expdocente, niveleduc, EXPINV y FORINV hasta obtener los resultados de la red neuronal, la cual arrojara el menor error de cada repetición.

La matriz mostrara los coeficientes que viajan en dirección entrada salida, así cada entrada de la función neuralnet quiere decir lo siguiente:

1. nn[1] call: la instrucción de llamado del modelo.
2. nn[2] response: extracción del valor de la variable respuesta.
3. nn[3] covariate: el valor de las variables extraída de la muestra utilizada.
4. nn[4] model.list: una lista que contiene el nombre de la variable respuesta (response) y las variables explicativas (covariates).
5. nn[5] err.fct: la función error.
6. nn[6] act.fct: la función activación.
7. nn[7] linear.output: nos dice si la salida es lineal.

8. nn[8] data: nos muestra la base de datos.

9. nn[9] net.result: una lista que contiene el resultado global de la red neuronal para cada repetición.

10. nn[10] weights: una lista que contiene los pesos ajustados de la red neuronal para cada repetición.

11. nn[11] startweights:una lista de los pesos iniciales de la red neuronal para cada repetición.

12. nn[12] generalized.weights:una lista que contiene los pesos generalizados de la red neuronal para cada repetición.

13. nn[13] result.matrix: una matriz que contiene el umbral alcanzado, las medidas necesarias, el error, el AIC y el BIC, y los pesos de cada repetición. Cada columna representa una repetición.

14. Hidden: número de capas ocultas.

Los pesos de conexión son parámetros desconocidos que deben de estimarse por un método de entrenamiento. El método más comúnmente utilizado es el de propagación hacia atrás (backpropagation), el cual se trató en proyecto de Titulo I.La idea básica es reducir el valor del error de la salida de la red. [Orallo et al., 2004]

### 8.1.3. Backpropagation

El algoritmo de aprendizaje backpropagation es el más comúnmente utilizado para entrenar un perceptron multicapa (*Multilayer Perceptron*, MLP); es decir, red neural que tiene una o más capas ocultas, la cual se utiliza para modelar un conjunto de datos cuando no es linealmente separable. Esta potencialidad del MLP dio origen a la regla de aprendizaje (algoritmo) de Retropropagación o propagación hacia atrás (backpropagation) [Orallo et al., 2004]. El algoritmo se trató en Proyecto de Título I .Lo que hace es cambiar los pesos iterativamente durante el entrenamiento con la finalidad de minimizar el error. El error entre las salidas reales de la red y la deseada se calcula y se utiliza para ajustar los pesos de la conexión. El procedimiento de ajuste es derivado por el método del descenso del gradiente (tratado en Proyecto de título I) y se utiliza para reducir la magnitud del error.[Orallo et al., 2004]

Se puede utilizar el número de capas ocultas que se quiera tomando en cuenta el tiempo de entrenamiento el cual puede ser excesivo para arquitecturas de muchas capas, además está demostrado que redes con una capa son capaces de aproximar cualquier función. [Orallo et al., 2004]

Con el siguiente código de R se estiman los pesos y la red neural.

```
nn <- neuralnet(p43~nombre+TIPOLOGI+INSTITUC+GEMELOS+SEXO+edad+expdocente+niveleduc+
                  EXPINV+FORINV,data=train, hidden=2,linear.output=FALSE,rep=3)
```

Lo cual produce la siguiente salida:

```
$result.matrix
                                  1               2              3
error                  27.509829165974 27.509396310167 27.50905918387
reached.threshold       0.009829324918  0.009396455449  0.00905931895
steps                  28.000000000000 23.000000000000 18.00000000000
Intercept.to.1layhid1   0.912446395449  1.499124437086  2.88221622429
TIPOLOGI.to.1layhid1    3.281972120942  0.061172998119  2.76865439519
INSTITUC.to.1layhid1    2.746494340908  1.150273939835  0.52433810769
SEXO.to.1layhid1        1.123857373861  0.922350734054  1.39693021603
edad.to.1layhid1        2.914426321744  1.209002500709  1.84758798495
expdocente.to.1layhid1  1.137709248824  1.660295827997  1.94363007792
niveleduc.to.1layhid1   1.416766657666  2.489061137167  1.56312775276
EXPINV.to.1layhid1      3.005118707680  1.447289425186  1.41452960489
FORINV.to.1layhid1      0.302374818887  3.833974153815  2.96287447725
Intercept.to.1layhid2   3.400976090703  1.559339439518  0.92972090919
TIPOLOGI.to.1layhid2    2.095670548104  2.434635884771 -0.45224934045
INSTITUC.to.1layhid2    1.938060610786  3.184976311107  2.00431981199
SEXO.to.1layhid2        3.677315520601  0.514505274300 -0.57505673420
edad.to.1layhid2        1.567110496515  2.212749121245  2.64282053072
expdocente.to.1layhid2  2.012938639463  1.398911045850 -0.25648489362
niveleduc.to.1layhid2   1.318543871225  2.820403641544  0.36229191774
EXPINV.to.1layhid2      1.146178570940  1.592780575764  1.93944807434
FORINV.to.1layhid2      0.435405930189  1.307968700174  1.88675497704
Intercept.to.p43        2.930962247339  1.555274223014  3.71857166927
1layhid.1.to.p43        2.441534745032  1.897918941259  3.18012562243
1layhid.2.to.p43        3.257253594677  5.221593720270  1.81262726256

attr(,"class")
[1] "nn"
```

Hay 3 repeticiones, las cuales están ordenadas según el error cometido por la red neuronal en cada entrenamiento (de menor a mayor), la segunda fila es el umbral alcanzado en la función de activación y la tercera fila son los escalones (vueltas o pasos) que necesita la red para alcanzar los pesos óptimos.

Por lo tanto observamos que tenemos varias redes con distintas tasas de error, al hacer tres repeticiones con el fin de obtener la mejor red, así representamos la mejor de ellas. Escogemos la tercera columna de la tabla porque es la que tiene el error más bajo la cual arrojó un error de 27.50905918387, graficamos y la figura 6 representa el modelo.

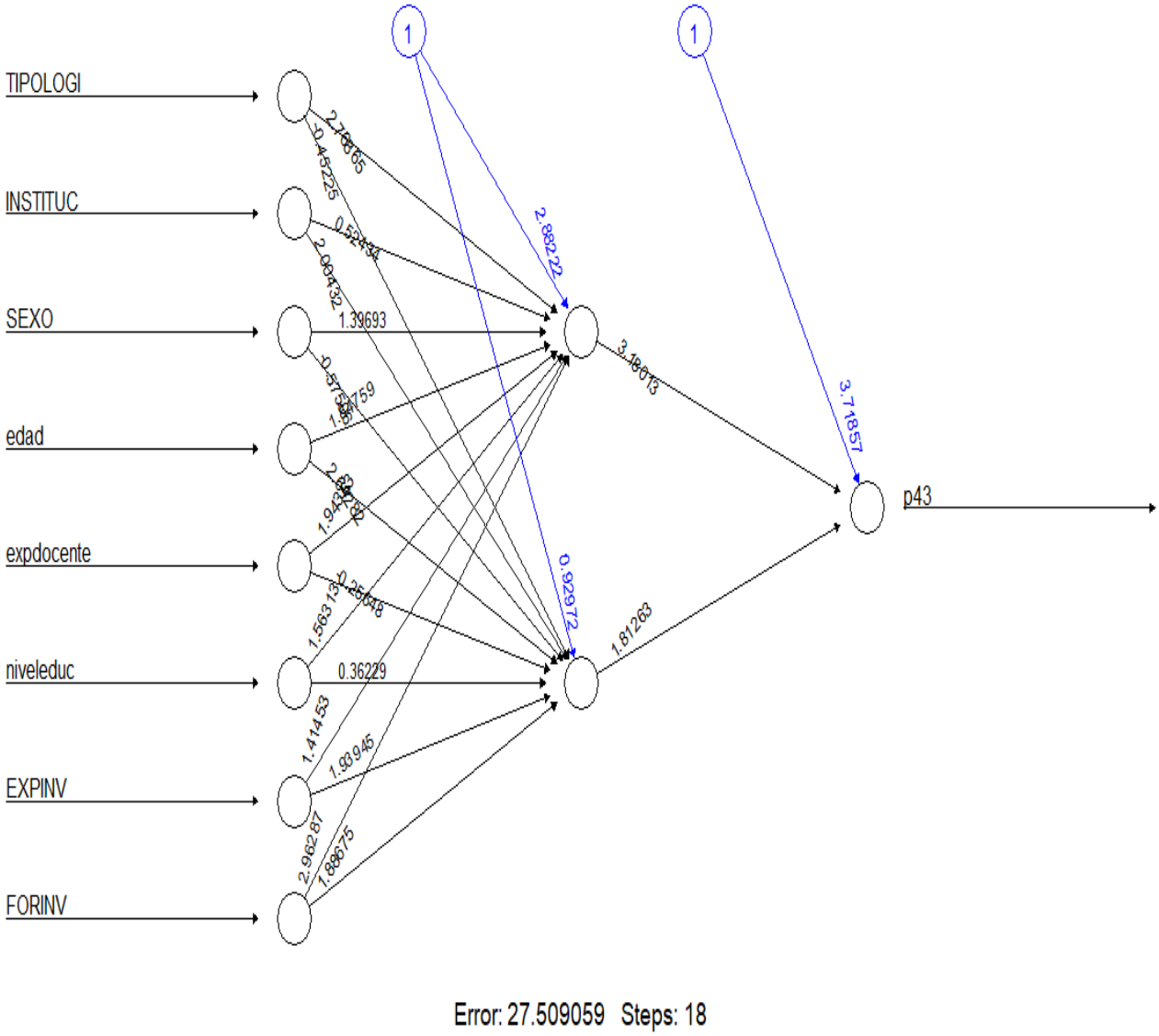


Figura 6: Mejor modelo red neuronal del impacto de investigación sobre la docencia.

Las líneas negras muestran las conexiones entre cada capa y los pesos de cada conexión, mientras que las líneas azules muestran el término de polarización añadido en cada paso.

La red es esencialmente una "caja negra" ya que pasamos unas variables de entrada por una capa oculta y obtenemos una salida, por lo que no podemos decir mucho sobre el ajuste, los pesos y el modelo. No sabemos lo que hace la red por dentro. Basta decir que el algoritmo de entrenamiento ha convergido y por lo tanto el modelo está listo para ser utilizado.

Así observamos que el proceso necesita 18 pasos.

Hay una capa oculta con dos neuronas ocultas y una capa de salida.

La utilización de una sola capa oculta es adecuada para la mayoría de los problemas de clasificación, pero cuando son numerosas las clases de salida se aconseja emplear al menos dos capas ocultas para producir un resultado más exacto [Kavzoglu and Mather, 2003] es por eso que se elige una capa oculta.

Los pesos estimados de la primera y segunda neurona oculta van desde 2.88221622429 a 0.92972090919 , respectivamente.

Los diez pesos estimados para las variables TIPOLOGI, INSTITUC, SEXO, edad, expodocente, niveleduc, EXPINV, FORINV que conducen a la primera neurona oculta son:

```
2.76865439519
0.52433810769
1.39693021603
1.84758798495
1.94363007792
1.56312775276
1.41452960489
2.96287447725 respectivamente.
```

Los diez pesos estimados para las variables TIPOLOGI, INSTITUC, SEXO, edad, expodocente, niveleduc, EXPINV, FORINV que conducen a la segunda neurona oculta son:

```
-0.45224934045
 2.00431981199
-0.57505673420
 2.64282053072
-0.25648489362
 0.36229191774
 1.93944807434
 1.88675497704 respectivamente.
```

Lo más interesante de los resultados anteriores es que de los pesos de la red neuronal se derivan fácilmente la importancia relativa de las variables de entrada.

Así observamos que precisamente las variables de mayor importancia para la primera neurona oculta son:FORINV, TIPOLOGI, expodocente, edad, niveleduc, EXPINV, SEXO, INSTITUC.

Para la segunda neurona oculta tenemos que las variables de mayor importancia son: edad, INSTITUC, EXPINV, FORINV, niveleduc, expodocente, TIPOLOGI, SEXO.

### 8.1.4. Evaluación del modelo.

Una vez entrenada nuestra red procedemos a evaluar la predicción de la red neuronal. Nuevamente utilizamos la matriz de confusión.

| | | **Real** | |
|---|---|---|---|
| | **de acuerdo** | **en desacuerdo** | **Total** |
| **Estimado** **de acuerdo** | 6 | 8 | 14 |
| **Estimado** **en desacdo** | 0 | 0 | 0 |
| **Total** | 6 | 8 | 14 |

Cuadro 6: Resultados de la matriz de confusión para modelo de Redes Neuronales.

La matriz de confusión presentada en el cuadro 6 indica que el modelo de redes neuronales estima 14 observaciones clasificadas como de acuerdo, de las cuales 6 estarían bien clasificadas y 8 se encontrarían mal clasificadas. Con estos datos se puede concluir que la precisión (observaciones bien clasificadas) del modelo en la matriz de confusión es de un 42.85 %, por lo tanto, el **Error de Predicción es de un 57.15 %**.

Para validar los resultados antes descritos y verificar el error de predicción se utilizó la técnica de validación cruzada.

### 8.1.5. Promedio del error de Redes Neuronales.

A continuación se muestra el resultado del promedio del error al realizar 10 iteraciones, resultando un **error promedio de predicción de un 27.15 %.**

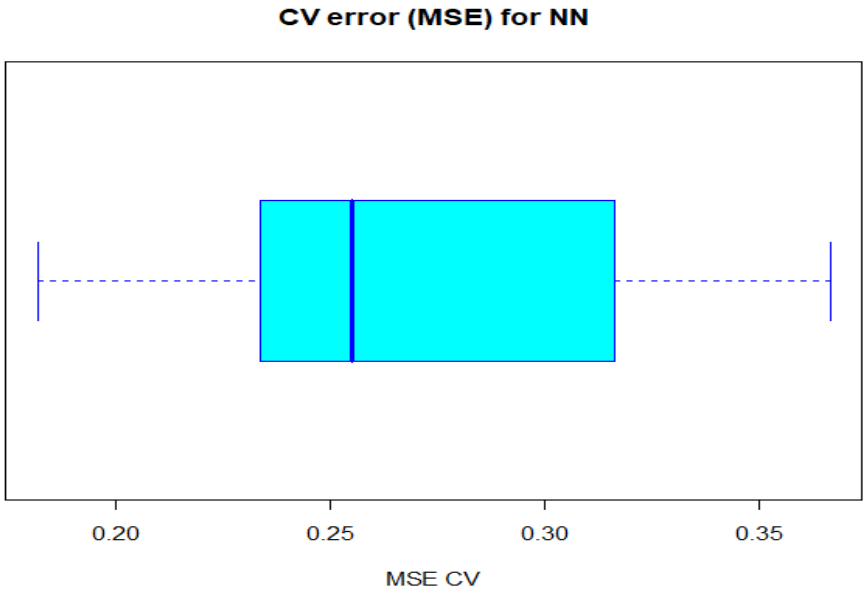


Figura 7: Promedio del error cuadratico medio para Redes Neuronales .

# 9. Aplicando Support vector machine.

## 9.1. Introducción.

Las máquinas de soporte vectorial (SVM, del inglés Support Vector Machines) pertenecen a la familia de los clasificadores lineales puesto que inducen separadores lineales o hiperplanos en espacio de características de alta dimensionalidad (introducidos por funciones núcleo o kernel) con un sesgo muy particular (maximización del margen). [Orallo et al., 2004]

La idea de maximizar el margen consiste en seleccionar el hiperplano separador que está a la misma distancia de los puntos más cercanos de cada clase. [Hastie et al., 2002]

Las máquinas de soporte vectorial son una técnica de aprendizaje automático muy popular y fácil de usar. Con las implementaciones actuales hoy en día son el estado de arte para problemas de clasificación. En problemas de clasificación queremos un modelo que nos permita separar datos de clases distintas o de múltiples clases. SVM busca una línea muy particular, esta es la línea que maximiza la distancia entre dos clases que no tengan elementos entre ellos. Los puntos que están en la frontera de la región de decisión son usados para definir esta línea, se les conoce como vectores de soporte, de ahí su nombre, por supuestos las matemáticas para encontrar esas líneas son intensas. [Orallo et al., 2004]

## 9.2. Técnicas de evaluación y selección de modelos.

Las técnicas de minería de datos presentadas anteriormente (árboles de decisión, redes neuronales y SVM) generan modelos por medio de la evidencia formada por un conjunto de datos. Luego nos preguntamos ¿Cómo se sabe qué modelo es lo suficientemente valido o bueno?. En estadística clásica se podía determinar mediante un análisis a la hipótesis nula.

Existen varias técnicas para seleccionar y evaluar modelos o hipótesis mediante la evidencia:

1. Evaluación mediante validación cruzada.
2. Evaluación por bootstrap.
3. Evaluación de hipótesis basada en costes.
4. Análisis ROC.
5. Evaluación de hipótesis basada en precisión y alcance.

Muchas veces es necesario evaluar de manera exacta la calidad de los modelos Por ejemplo si en la detección de células cancerígenas se aplica un modelo el cual comete un error en la predicción esto lleva a importantes consecuencias por tal razón es importante saber con exactitud el nivel de precisión del modelo aprendido.

Si se utiliza el conjunto de entrenamiento para evaluar la calidad de un modelo se puede estar equivocado porque se premia a los modelos que se ajustan más al conjunto de entrenamiento y se favorecen los modelos que sobreajustan el conjunto de entrenamiento y no generalizan para todos. [Orallo et al., 2004]

Los datos que forman la evidencia se separan en dos subconjuntos, los de **entrenamiento** (*training*) que se utilizan para el aprendizaje de las hipótesis y el de **prueba** (*test*) que se emplea para calcular el error de muestra de la hipótesis construida con los datos de entrenamiento.

Cuando utilizamos los dos subconjuntos de datos independientes para aprender y evaluar la hipótesis, se permite resolver el problema de premiar el sobreajuste en la evaluación de la hipótesis.

El **sobreajuste** se define como la situación en la que el *modelo da mucho mejores resultados para el conjunto de entrenamiento que para el conjunto test.* [Orallo et al., 2004]

Existe otra técnica de evaluación basada en costes. La cual evalúa el coste de los errores cometidos por un modelo. En este contexto, el mejor modelo es el que comete errores con menor coste asociado, no el modelo que cometa menor número de errores. [Orallo et al., 2004]

## 9.3. Evaluación basada en coste y matriz de confusión.

La precisión de una hipótesis (porcentaje de error de la hipótesis con respecto a una función objetivo f) generalmente no es la mejor medida para evaluar la calidad de un determinado modelo o un determinado algoritmo de aprendizaje. La calidad de un determinado modelo se mide en términos de minimización de costes en vez de minimización de errores.

Por ejemplo en un hospital se desea mejorar los criterios de admisión utilizando técnicas predictivas de minería de datos. Así cuando llega un paciente se le asignan tres posibles destinos dependiendo de las condiciones del paciente:

1. UCI: Unida de Cuidados Intensivos.
2. Observación: Ingreso en Observación.
3. Salida: Salida con alta.

Con probabilidades de cada uno de los casos de 0.5, 13, y 86.5 % respectivamente.

Si se aprende un modelo de predicción con alguna técnica de aprendizaje, un modelo que siempre enviase a un paciente a casa obtendría una precisión del 86,5 por ciento de los casos. Desde el punto de vista de la precisión el modelo es válido, sin embargo, el modelo sería inútil y peligroso ya que se enviarían a casa pacientes en estado grave.

Es más conveniente utilizar técnicas de aprendizaje que obtienen modelos que minimicen los costes de aplicación. Considérese que se conocen aproximadamente los costes de clasificación errónea. La manera de representar estos costes en problemas de clasificación es en una matriz de costes.

**Definición 1** ***Matriz de coste:*** *Matriz en la cual se representan los costes de todas las posibles combinaciones que se pueden dar entre la clase predicha (estimada) y la real. Por lo tanto para un problema de c clasificadores, la matriz debe ser de orden c x c.*

A continuación se muestra la matriz de costes para el ejemplo del problema de las salas de urgencias del problema anterior.

| | **Real** | | |
|---|---|---|---|
| **Predichas** | Salida | Observación | UCI |
| Salida | 0 | 5.000 | 500.000 |
| Observación | 300 | 0 | 50.000 |
| UCI | 800 | 500 | 0 |

Cuadro 7: Matriz de costes (en euros) para el ejemplo del problema de las salas de urgencias.

Interpretación.

Si el sistema predice que un paciente debe ir a la UCI, cuando en realidad no necesita tratamiento tiene un coste de 800. Notar que todos los costes de la diagonal (es decir de aciertos) pueden expresar costes positivos y negativos.

Si se dispone de una matriz de costes se pueden estimar los costes de los clasificadores por medio de una matriz de confusión.

**Definición 2** ***Matriz de confusión:*** *Es una manera detallada de informar cómo distribuye los errores un determinado clasificador.*

A continuación se muestra la matriz de confusión para una evidencia formada por 100 casos (conjunto de test) para el problema de la sala de urgencias.

| | **Real** | | |
|---|---|---|---|
| **Predichas** | salida | Observación | UCI |
| Salida | 71 | 3 | 1 |
| Observación | 8 | 7 | 1 |
| UCI | 4 | **2** | 3 |

Cuadro 8: Matriz de confusión para el problema de la sala de urgencia.

Cada celda de la matriz de confusión enumera, para cada clase real, el número de casos en los cuales el clasificador ha predicho una clase. Por ejemplo el clasificador ha enviado dos pacientes a la UCI, de los que deberían haberse enviado a observación. De la matriz de confusión se puede estimar la precisión de un clasificador directamente, dividiendo el número de aciertos (casos en la diagonal) entre el número total de casos. Para este clasificador la precisión sería del 81 %.

Estimar el coste de un clasificador es bastante sencillo para una determinada evidencia si se disponen de las matrices de coste y de confusión:

$$Coste = \sum_{\substack{1 \leq i \leq n \\ 1 \leq j \leq n}} C_{ij} M_{ij}$$

Donde $C_{ij}$ expresa la posición $i, j$ de la matriz de coste, y $M_{ij}$ la posición $i, j$ de la matriz de confusión. Por ejemplo el coste del clasificador asociado a la anterior matriz de confusión utilizando la matriz de coste para el problema de la sala de urgencias sería de 571.600 euros. En cambio, el coste del clasificador que siempre da el alta sería de 2.560.000 euros con una precisión del 83 por ciento. En el ejemplo vemos que un clasificador con más errores puede tener mucho menos coste.

## 9.4. Análisis ROC.

Por desgracia, no siempre se dispone de una matriz de coste que permita estimar o adaptar el aprendizaje en un determinado contexto de coste. Muchas veces la matriz de coste solo se conoce durante el tiempo de aplicación y no durante el aprendizaje, generalmente por que los costes varían frecuentemente o son independiente del contexto.
[Orallo et al., 2004]

Cuando no tenemos la matriz de coste lo que se hace es aprender un conjunto de clasificadores y seleccionar el que mejor se comporte para circunstancias o contextos de costes determinados a posteriori. Para ello usa la técnica de análisis ROC. (Receiver Operating Caracteristic) la cual provee herramientas que permiten seleccionar el subconjunto de clasificadores que tienen un comportamiento optimo general. De igual manera el análisis ROC permite evaluar clasificadores de manera más independiente y completa a la clásica precisión.
[Orallo et al., 2004]

En análisis ROC se utiliza normalmente para problemas en los que se tienen dos clases (se suelen denominar clase positiva y clase negativa), y para este tipo de problemas se utiliza la matriz de confusión empleándose la siguiente notación:

| | **Real** | **Real** |
|---|---|---|
| **Predicha** | Criterio 1 | Criterio 2 |
| Criterio 1 | True Positives (TP) | False Positives(FP) |
| Criterio 2 | False Negatives (fN) | True Negatives (TN) |

Cuadro 9: Notación para la matriz de confusión.

Concretamente se utilizan estos valores normalizados por columna, con la siguiente notación:

1. True Positive Rate: $TPR = \dfrac{TP}{(TP + FN)}$
2. False Negative Rate:$FNR = \dfrac{FN}{(TP + FN)}$
3. False Positive Rate:$FPR = \dfrac{FP}{(FP + TN)}$
4. True Negative Rate: $TNR = \dfrac{TN}{(FP + TN)}$

Con estos valores se completa la matriz de confusión normalizada:

| | **Real** | **Real** |
|---|---|---|
| **Predicha** | Criterio 1 | Criterio 2 |
| Criterio 1 | TPR | (FPR) |
| Criterio 2 | fNR | (TNR) |

Cuadro 10: Matriz de confusión Normalizada.

Donde, TPR=1-FNR, y FPR=1-TNR. Así podemos seleccionar solo dos de los ratios y construir un espacio bidimensional (llamado espacio ROC), limitado por los puntos (0,0) y (1,1). El análisis se basa en representar cada clasificador como un punto en el espacio ROC. En el análisis ROC se utilizan los valores TPR para el eje de las y, mientras que FPR para el eje de las x.

Por ejemplo dado un clasificador con la siguiente matriz de confusión:

| | **Real** | **Real** |
|---|---|---|
| **Predicho** | 30 | 30 |
| **Predicho** | 20 | 70 |

Cuadro 11: Ejemplo matriz de confusión .

Así de la matriz de confusión obtenemos los valores TPR=0,6 y FPR=0,3 con los cuales se grafica el punto de corte. La figura 8 muestra un ejemplo de un clasificador en el espacio ROC.

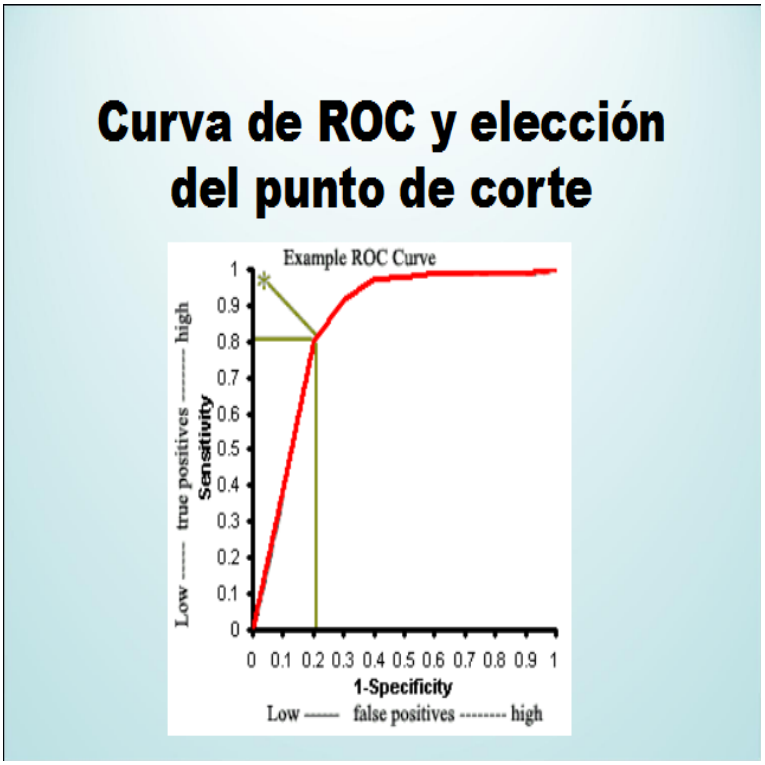


Figura 8: Ejemplo de clasificador en espacio ROC.

Los puntos (0,0) y (1,1) corresponden, respectivamente, el clasificador que predice todo como clase positiva, y el clasificador que asigna todo como clase negativa. Ambos clasificadores se conocen como clasificadores triviales.

Representar un clasificador en un espacio ROC da lugar a la siguiente propiedad :
[Provost et al., 1997]
Dado un conjunto de clasificadores, se pueden descartar los clasificadores que no caigan dentro de la superficie convexa (área bajo la curva) formada por todos los clasificadores, junto con los puntos (0,1) y (1,0) y (0,0).

Por lo tanto, desde el punto de vista del aprendizaje sensible al coste, el mejor sistema de aprendizaje será aquel que produzca el conjunto de clasificadores con mayor área bajo la superficie convexa o AUC (*Area Under the ROC Curve*).

## 9.5. Resultados.

Primero chequearemos si las clases son linealmente separable para saber si el problema de clasificación binaria es perfectamente separable y así utilizar los SVM con margen máximo (duro) o SVM con margen suave (blando).

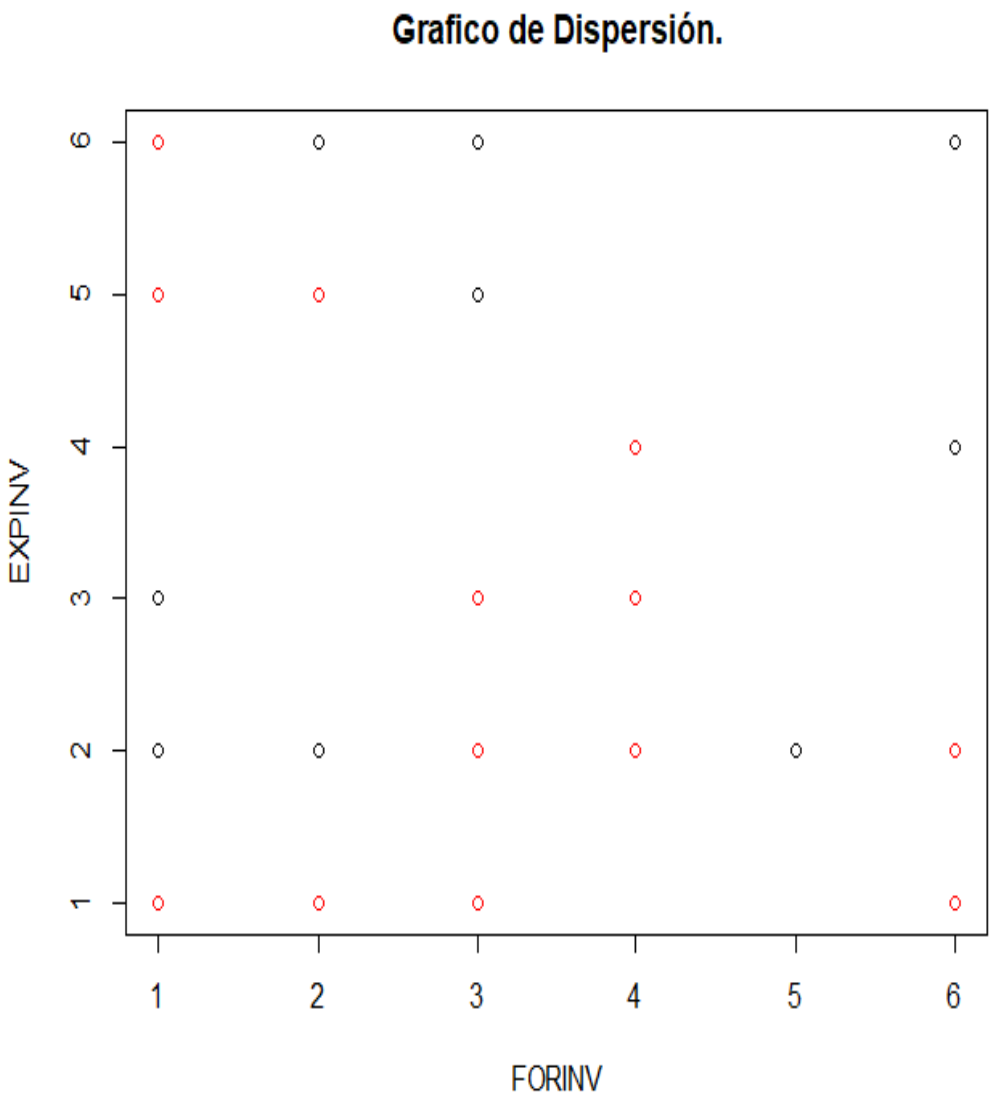


Figura 9: Diagrama de dispersión de todas las variables independientes EXPINV y FORINV.

Como se puede observar en la figura 9 las variables independientes EXPINV y FORINV son difíciles de separar por un hiperplano, de hecho, se trata de dos tipos de clases no separables. El problema de determinar el clasificador de margen máximo duro no se puede aplicar en esta separación. Por lo tanto se aplicara el clasificador de margen máximo blando con parámetro $C = 1$. [Han et al., 2011]

Los elementos de clase 1 se denotan con un círculo negro mientras que los de clase 2 con un círculo rojo.

### 9.5.1. SVM lineal.

Para calcular la regla de clasificación SVM lineal con C=1, se usa la función **svm** del paquete **e1071**. El primer argumento y∼. indica que la variable dependiente y (que debe ser necesariamente un factor, para nuestro caso 1 y 2) se desea predecir en términos del resto de las variables. La sintaxis es similar a la que utilizaríamos para ajustar un modelo lineal o un modelo de regresión logística. El segundo argumento indica el fichero en el que están las variables que vamos a usar. El argumento **kernel** corresponde al núcleo que representa el producto escalar que queremos utilizar. La opción linear corresponde a $k(x, y) = x'y$. Existen varias tipos de kernel como por ejemplo, polinomial, gaussiana o RBF, exponencial, radial, sigmoidal, etc. Todos mencionados en Proyecto de Titulo I. El argumento **cost** determina la penalización que ponemos a los errores de clasificación.Cost nos permite especificar el costo de una violación al margen. Cuando el argumento cost es pequeño, entonces los márgenes serán anchos y muchos vectores de soporte estarán en el margen o violarán el margen. Cuando el argumento cost es grande, entonces los márgenes serán estrechos y habrán pocos vectores de soporte en el margen o violando el margen. Con el fin de estimar la probabilidad de clasificar erróneamente una observación se utiliza validación cruzada, dividiendo la muestra en dos partes. Ello se consigue fijando **cross=2**. Finalmente, **scale=FALSE** se usa para datos no estandarizados (por defecto, sí se estandarizan). [James et al., 2014]

Podemos obtener cierta información básica sobre el SVM lineal utilizando el comando summary ().

```
Call:
svm(formula = p43 ~ ., data = train, kernel = "linear", type = "C-classification", scale = FALSE)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  1
      gamma:  0.125

Number of Support Vectors:  143

 ( 73 70 )
```

La salida anterior nos dice que se utilizó un kernel de tipo lineal con cost=10, y que Hay 143 vectores de soporte ,73 para la clase de tipo 1 y 70 para la clase de tipo 2.

La figura 10 muestra las dos zonas en las que la superficie óptima de decisión divide al plano $\Re^2$. Los vectores de soporte se denotan con una cruz, mientras que los elementos que no son vectores de soporte aparecen representados con un círculo. Para representar cada una de las dos clases, se emplea un código de color rojo para los elementos de la clase 1 y negro para los de la clase 2.

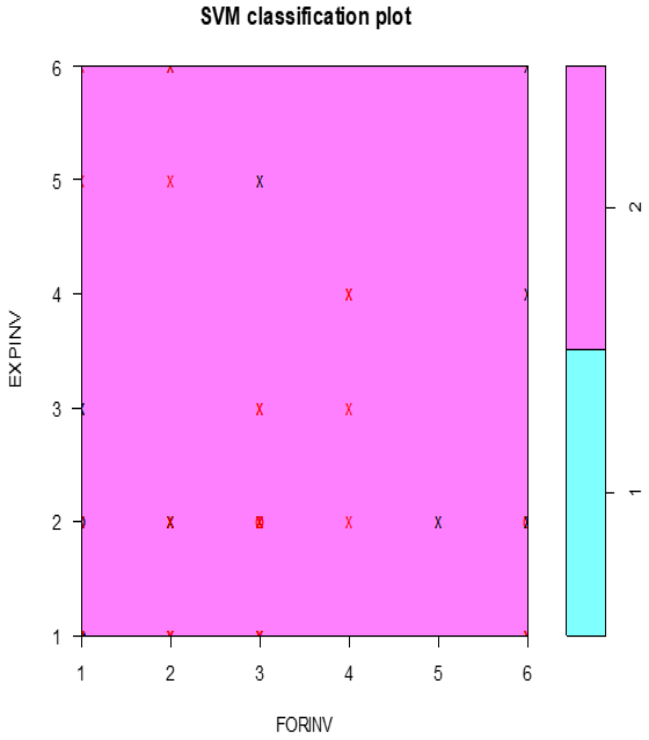


Figura 10: SVM con kernel lineal.

Podemos determinar las identidades de los vectores de soporte de la siguiente manera.

```
modelo1$index
```

obteniendo así la siguiente salida.

```
  [1]   3   4   5   6   7   8  10  11  13  16  17  19  24  25  27  31  33  35  36  43  45
       47  49  52  60  62  63  65  66
 [30]  68  70  71  74  76  78  80  84  85  87  90  91  93  94  98 100 101 102 107 109 110
      114 115 116 122 125 126 131 132
 [59] 133 135 136 137 140 143 144 145 146 149 150 154 155 157 161   2  12  15  20  21  22
       26  28  29  30  32  34  38  39
 [88]  40  41  42  44  50  53  54  55  58  59  61  64  67  69  72  73  75  79  81  82  83
       88  89  92  95  97  99 103 104
[117] 105 106 108 111 112 117 118 119 120 121 123 124 127 128 130 134 139 141 142 147 148
      151 152 153 156 158 159
```

El paquete e1071 incluye la función tune (), para realizar la validación cruzada. Por defecto, tune () realiza diez veces la validación cruzada en un conjunto de modelos de interés. Para utilizar esta función, se pasa información relevante sobre el conjunto de modelos que se están considerando.

El siguiente comando indica que queremos comparar SVMs con kernel lineal, usando un rango de valores del parámetro cost y gamma.

```
obj1 <- tune.svm(p43~., data =train,kernel="linear", gamma = 2^(-1:1), cost = 2^(1:3))
```

Podemos acceder fácilmente a los errores de validación cruzada para cada uno de estos modelos usando el comando summary ():

```
Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 gamma cost
   0.5    2

- best performance: 0.4816418

- Detailed performance results:
  gamma cost     error dispersion
1   0.5    2 0.4816418  0.1267942
2   1.0    2 0.4816418  0.1267942
3   2.0    2 0.4816418  0.1267942
4   0.5    4 0.4816635  0.1267786
5   1.0    4 0.4816635  0.1267786
6   2.0    4 0.4816635  0.1267786
7   0.5    8 0.4819380  0.1265528
8   1.0    8 0.4819380  0.1265528
9   2.0    8 0.4819380  0.1265528
```

Vemos que los resultados de costo = 2 y gamma = 0.5 obtuvieron la menor tasa de error de validación cruzada.

La función tune () almacena el mejor modelo obtenido,se puede acceder de la siguiente manera:

```
best.mod1=obj1$best.model
summary(best.mod1)

Call:
best.svm(x = p43 ~ ., data = train, gamma = 2^(-1:1), cost = 2^(1:3), kernel = "linear")


Parameters:
   SVM-Type:  eps-regression
 SVM-Kernel:  linear
       cost:  2
      gamma:  0.5
    epsilon:  0.1


Number of Support Vectors:  144
```

### 9.5.2. Evaluación del modelo.

Se procede a evaluar la predicción del modelo1. Para ello, se utilizará la matriz de confusión, los resultados de la matriz van cambiando dado que se obtienen del conjunto de validación pero las diferencias no son grandes.

La matriz de confusión presentada en el cuadro 12 indica que el modelo1 estima 8 observaciones clasificadas como de acuerdo, de las cuales 5 estarían bien clasificadas y 3 se encontrarían mal clasificadas y de las 10 observaciones estimadas en la clase en desacuerdo, 5 se encontrarían bien clasificadas y 5 estarían mal clasificadas. Con estos datos se puede concluir que la precisión (observaciones bien clasificadas) del modelo en la matriz de confusión es de un 55.55 %, por lo tanto, el ***Error de Predicción es de un 44.45 %.***

| | | **Real** | |
|---|---|---|---|
| | | **de acuerdo** | **en desacuerdo** | **Total** |
| **Estimado** | **de acuerdo** | 5 | 3 | 8 |
| | **en desacdo** | 5 | 5 | 10 |
| | **Total** | 10 | 8 | 18 |

Cuadro 12: Resultados de la matriz de confusión para modelo1 con kernel linel.

De igual forma se crearon modelos (estimación de parámetros) SVM de tipo radial, polinomial, sigmoidal, y gausiano. Para validar los resultados y verificar el error de predicción, se utilizó nuevamente la técnica de validación cruzada. A continuación se muestra el cuadro número 13 con los siguientes resultados:

| Modelo | Kernel | Precisión | Errores de predicción | Promedio ( % de predicción) | Promedio del Error de Predicción |
|---|---|---|---|---|---|
| modelo1 | lineal | 55.55 | 44.45 | 62.25 | **37.74** |
| modelo2 | radial | 66.66 | 33.34 | 54.21 | 45.79 |
| modelo3 | polinomial | 50.00 | 50.00 | 53.14 | 46.86 |
| modelo4 | sigmoidal | 44.44 | 55.56 | 57.10 | 42.90 |
| modelo5 | gausiano | 66.66 | 33.34 | 57.91 | 42.09 |

Cuadro 13: Promedio del error de Predicción para distintos modelos de SVMs.

Como se observa en el cuadro 13, si bien los modelos 2 y 5 con kernel de tipo radial y gausiano respectivamente presentan un porcentaje de acierto superior, lo que indica observaciones bien clasificadas para ambos modelos, el promedio del error de predicción es muy alto comparado con el modelo1 con kernel radial. ***Por lo tanto elegimos el modelo con kernel radial como mejor***

***modelo de predicción con un promedio del error de predicción de un 37.74 %.***

A continuación se muestra gráfico % de predicción de modelo1 con kernel lineal (mejor modelo) y todos los gráficos de % de predicción para los svm con distinto kernel.

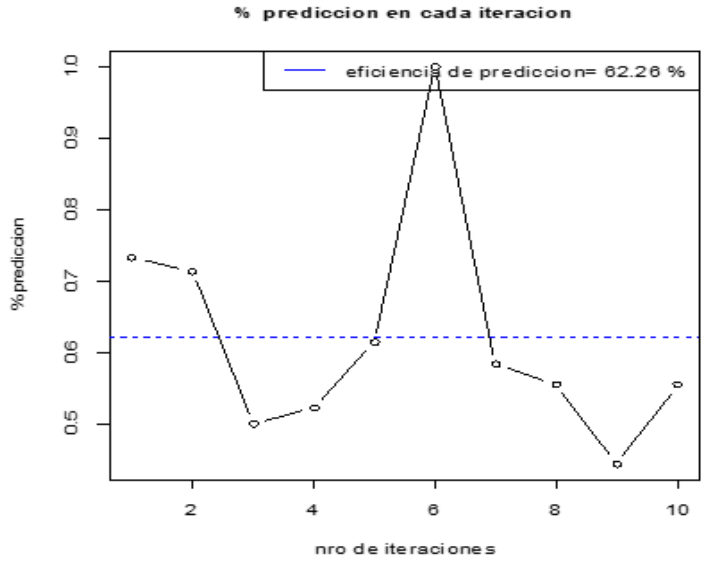


Figura 11: Gráfico promedio de la eficiencia general del modelo predictivo con kernel lineal

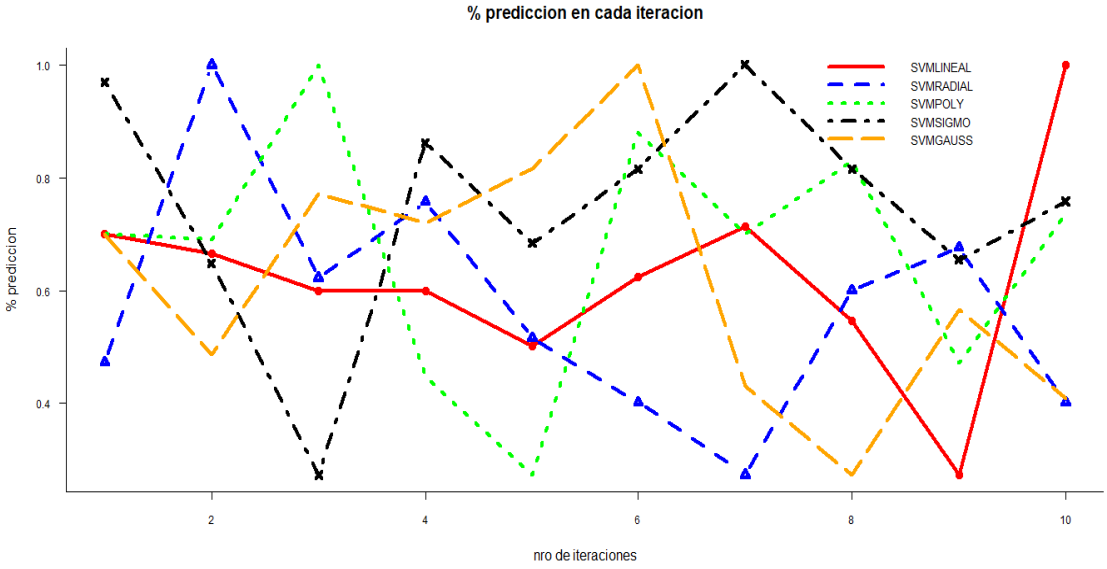


Figura 12: Gráfico de la eficiencia general de modelos predictivos con distinto kernel.

Se puede concluir que el método estadístico de clasificación binario Support Vector Machine, es una forma eficaz y sencilla de solucionar problemas de clasificación de dos clases.

En segundo lugar, se observa que el rendimiento de una SVM está directamente relacionado con los parámetros de diseño fijados, los cuales corresponden al parámetro C de penalización, el tipo de función Kernel a utilizar y los parámetros de la misma.

Se logró identificar el mejor model, el tipo radial con un promedio del error de prediccion de un 37,74 %.

# 10. Conclusiones.

Durante el presente Proyecto de Título II se estudió el problema de que si se estaba en desacuerdo o de acuerdo en el impacto de la investigación sobre la docencia desde un mirada de la minería de datos aplicando tres técnicas de predicción como árboles de decisión, redes neuronales y soporte de vectores máquinas. Además se desarrollaron modelos de clasificación para cada técnica.

La siguiente tabla presenta una comparación entre los tres modelos obtenidos:

| | Tecnicas De Predicción | | |
|---|---|---|---|
| Factor a Comparar | Árboles de Desición | Redes Neuronales | Support Vector Machine |
| Validación Cruzada: Promedio del Error de Predicción. | 38.26 | 27.15 | 37.74 |

Cuadro 14: Elección mejor técnica de predicción.

El objetivo del presente trabajo era crear un modelo de Árbol de Decisión, Redes Neuronales y de SVM que predijeran si se estaba de acuerdo o en desacuerdo en el impacto de la investigación sobre la docencia, y decidir cuál es el mejor modelo de predicción, escogiendo aquel que obtuviera un menor Promedio del Error de Predicción, obteniendo dicho promedio por la técnica de Validación Cruzada, al mirar el cuadro 14, se decide por el modelo de Redes Neuronales ya que presenta un menor error.

En este proyecto se estudiaron tres técnicas diferentes de predicción, sin embargo dada la cantidad de variables elegidas e introducidas en los modelos se piensa que una mejor elección de las variables podría ayudar a disminuir los promedios de los errores de predicción de los modelos.

# 11. Referencias

[Dreyfus, 2005] Dreyfus, G. (2005). Neural networks: methodology and applications. Springer Science & Business Media.

[Han et al., 2011] Han, J., Pei, J., and Kamber, M. (2011). Data mining: concepts and techniques. Elsevier.

[Hastie et al., 2002] Hastie, T., Tibshirani, R., and Friedman, J. (2002). The elements of statistical learning: Data mining, inference, and prediction. Biometrics.

[James et al., 2014] James, G., Witten, D., and Hastie, T. (2014). An introduction to statistical learning: With applications in r.

[Kavzoglu and Mather, 2003] Kavzoglu, T. and Mather, P. M. (2003). The use of backpropagating artificial neural networks in land cover classification. International journal of remote sensing, 24(23):4907–4938.

[Montaño Moreno et al., 2002] Montaño Moreno, J. J. et al. (2002). Redes neuronales artificiales aplicadas al análisis de datos.

[Orallo et al., 2004] Orallo, H., RAMIREZ, J., QUINTANA, C. R., Orallo, M. J. H., Quintana, M. J. R., and Ramírez, C. F. (2004). Introducción a la Minería de Datos. Pearson Prentice Hall,.

[Provost et al., 1997] Provost, F. J., Fawcett, T., et al. (1997). Analysis and visualization of classifier performance: comparison under imprecise class and cost distributions. In KDD, volume 97, pages 43–48.

# 12. Anexo. Códigos en R.

## 12.1. Árboles de Decisión.

```
###################### Árbol de desición CON RPART ##############
# El paquete \rpart" utiliza el algoritmo CART + Pruning (poda).

# Cargar e instalar las siguientes librerias.
# install.packages('rpart')
library(rpart)
library(ROCR) # Para curva ROC.

# Leer datos desde excel.
datos<-file.choose()
datosfinal<-read.csv2(datos,h=T)

# Vista de los datos en R.
View(datosfinal)

# codificación para variable dependiente u objetivo.
datosfinal$p43<-ifelse(datosfinal$p43==1 | datosfinal$p43==2 | datosfinal$p43==3,1,2 )

# veo mi variable dependiente codificada.
View(datosfinal$p43)

# veo mis datos con mi variable codificada.
```

```
View(datosfinal)

# Seleccionamos muestra para evitar problemas de overfitting(sobre ajuste de los datos)
dato_app<-sample(1:179,size=100,replace=FALSE)

# Entrena el modelo.
entrenamiento<-datosfinal[dato_app,]

# conjunto de validación. Datos no visto por el algoritmo para ver
# como huviese sido en una implementación real.
test<-datosfinal[-dato_app,]

#####################################
# se crean los Árboles de Decisión de manera aleatoria,
# utilizando el conjunto de entrenamiento y el método de clasificación
# ya que los Árboles de Decisión presentan dos forma, árboles de clasificación
# y árboles de regresión.
# Con los datos test evaluamos o analizamos el modelo generado.

# introducimos nuestro modelo.
# genero mi modelo.
tree1datosfinal<-rpart(p43~.,data=entrenamiento,method="class")
tree1datosfinal

# Creamos el árbol teórico, otra forma.
print(tree1datosfinal)

# El comando summary (nombre del árbol creado) entrega un resumen completo del árbol,
# donde muestra los cortes de los nodos, el valor del parámetro de complejidad por el que
# se poda el mismo nodo, las variables sustitutas con sus respectivos índices de mejora de Gini.

summary(tree1datosfinal) # predicciones y validación cruzada de mi base de entrenamiento.

# generamos el gráfico del árbol que nos ayudara a interpretar la información.
par(mar=rep(0.1, 4))
plot(tree1datosfinal)
text(tree1datosfinal, cex=0.5)

# cargar rpart.plot
# me da un gráfico mas hermoso.
library(rpart.plot)
rpart.plot(tree1datosfinal)

# Para podar el árbol se necesita el valor del CP, el cual se obtiene por medio de:

printcp(tree1datosfinal)

plotcp(tree1datosfinal)

# Una vez obtenido se realiza la poda del Árbol
# en cp colocamos el que tenga menor xerror.
```

```
tree1datosfinalb<-prune(tree1datosfinal, cp=0.040816)
tree1datosfinalb

# Gráfico árbol podado.
rpart.plot(tree1datosfinalb)

# A continuación, sobre el conjunto de validación, test,
# se realiza la predicción y se obtiene el resumen de esta:
pred <- predict(tree1datosfinal,test,type="class")
pred
summary(pred)

# matriz de confusión.
mc <- table(pred,test$p43, dnn=c("Estimado", "Real"))
mc

# Media del error de predicción por medio de la tecnica de Validación Cruzada.

# FOLDS
#-----------------------------------------------------
Folds<-10
datosfinal$kfold<-sample(1:Folds, nrow(datosfinal),replace=T)

# Modelos
#-----------------------------------------------------
Iter<-data.frame(iteracion=NULL, aciertos=NULL)
for(i in 1:Folds)
{
Test          <-subset(datosfinal, kfold==i)
Entrenamiento <-subset(datosfinal,!kfold==i)
Modelo        <-rpart(p43 ~.,data=Entrenamiento,method="class")

Prediccion    <-predict(Modelo, Test, type="class")
MC            <-table(Test[,"p43"],Prediccion)

Aciertos      <-MC[1,1]/(MC[1,1] + MC[2,1])
Iter          <-rbind(Iter,data.frame(Iter=i,acierto=Aciertos))
}


# Gráfico.
promedio <-format(mean(Iter$acierto, na.rm=TRUE)*100,digits=4)
plot(Iter,type="b",main="% prediccion en cada iteracion",
     cex.axis=1,cex.lab=1, cex.main=1,
     xlab="nro de iteraciones",ylab="% prediccion")
abline(h=mean(Iter$acierto),col="blue",lty=2)
legend("topright",legend=paste("eficiencia de prediccion=",promedio,"%"),
       col="blue", lty=1,lwd= 1, cex=1,bg=NULL)

##############################################################################
```

### 12.1.1. Redes neuronales.

```
# Leer datos desde excel
datos<-file.choose()
datosfinal<-read.csv2(datos,h=T)

# Vista de los datos en R.
View(datosfinal)

# Codificación.
datosfinal$p43<-ifelse(datosfinal$p43==1 | datosfinal$p43==2 | datosfinal$p43==3,1,2 )
View(datosfinal$p43)

View(datosfinal) # ahora aparece codificada mi variable respuesta

# Particiones de entrenamiento y validación

# 1) Obtenemos vector con indices aleatorios en fold.test.Así podemos hacer
    con una muestra la fase de aprendizaje del modelo, y con otra la fase de ejecución.
# 2) Para crear nuestra partición de entrenamiento y de validación usaremos
#   los índices anteriores, almacenados en la variable fold.test
# 3) Con la expresión datosfinal[fold.test, ] estamos tomando del data.frame
#   original las  columnas de las filas  cuyos índices contiene fold.test.
#   La expresión datosfinal[-fold.test, ] es similar,
#   pero tomando las filas cuyos índices no están en el vector fold.test.
#   De esta forma obtenemos dos conjuntos disjuntos de ejemplos, uno en la variable train y otro
#   en la variable test. Ambas son objetos data.frame, con las mismas  columnas que datosfinal.


fold.test <- sample(nrow(datosfinal), nrow(datosfinal) / 3)  # Muestra
test <- datosfinal[fold.test, ]
train <- datosfinal[-fold.test, ]

# Entrenamiento (ESTIMACION) de la red neuronal

nn<-neuralnet(p43~TIPOLOGI+INSTITUC+SEXO+edad+expdocente+niveleduc+EXPINV+
              FORINV,err.fct = "sse",data=train,hidden=2,linear.output=FALSE,rep=3)
nn

summary(nn)[5]


print(nn)
nn$result.matrix       # otra forma de imprimir las tres redes neuronales.

plot(nn, rep = "best") # "rep" me elige la mejor red


# A la modelización con redes neuronales siempre se le ha achacado
# un comportamiento de \caja negra",
# nosotros pasamos unas variables de entrada por una capa oculta y obtenemos una salida.
# No hay parámetros ni inferencia sobre los mismos,no sabemos lo que hace la red por dentro.
```

```
# los pesos de la red pueden ser utilizados para determinar cómo influye una variable
# en el modelo.

names(nn)
nn[13]

########################## Validación Cruzada. ##############################

### cargar e instalar neuralnet
library(neuralnet)

# Leer datos desde excel.
datos<-file.choose()
datosfinal<-read.csv2(datos,h=T)

# codificación para mi variable dependiente u objetivo.
datosfinal$p43<-ifelse(datosfinal$p43==1 | datosfinal$p43==2 | datosfinal$p43==3,1,2 )

View(datosfinal)


#####   Particiones de entrenamiento y validación de la red neuronal #############

# 1)obtenemos el vector indices en index.Así podemos hacer con una muestra
  la fase de aprendizaje del modelo, y con otra la fase de ejecución.

# 2) Para crear nuestra partición de entrenamiento y de validación usaremos
#    los índices que vienen acontinuacion, almacenados en la variable index.

# 3) Con la expresión datosfinal[index, ] estamos tomando del data.frame
#    original las  columnas de las filas cuyos índices contiene index.

# 4) La expresión datosfinal[-index, ] es similar,
#    pero tomando las filas cuyos índices no están en el vector index.
#    De esta forma obtenemos dos conjuntos disjuntos de ejemplos, uno en la variable
#    train y otro en la variable test. Ambas son objetos data.frame, con las mismas
#    columnas que datosfinal.


index<- sample(nrow(datosfinal),.9* nrow(datosfinal)# Se toma una muestra de los datos
train <- datosfinal[index,]                  # datos para generar y entrenar la red.
test <- datosfinal[-index,]                  # datos para validar la red.


# Scaling data for the NN (normalizacion de los datos).

# Es una buena práctica normalizar los datos antes de entrenar una red neuronal,
# es muy importante  este paso: dependiendo del conjunto de datos, evitar la
# normalización puede conducir a resultados erróneos o a un proceso de entrenamiento
# muy difícil (la mayoría de las veces el algoritmo no convergerá antes del número de
# iteraciones máximas permitidas). Se puede  elegir diferentes métodos para escalar
# los datos (normalización z, escala min-máx, etc.). Se utilizó el método min-máx y
```

```
# escalar los datos en el intervalo [0,1]. Por lo general, la escala en los
# intervalos [0,1] o [-1,1] tiende a dar mejores resultados.
# Por lo tanto, escalamos y dividimos los datos antes de seguir adelante:

maxs <- apply(datosfinal, 2, max)
mins <- apply(datosfinal, 2, min)

# Datos en 0 y 1
# Hay que tener en cuenta que scaled devuelve una matriz que necesita ser
# introducida en un archivo data.frame.

scaled <- as.data.frame(scale(datosfinal, center = mins, scale = maxs - mins))


# Train-test split
train_ <- scaled[index,]  # entrena la red.
test_ <- scaled[-index,]  # test_ se ocupan para validar el modelo (prediccion de datos nuevos)

# NN training( entrenamiento de la red neuronal)

n <- names(train_)
f <- as.formula(paste("p43 ~", paste(n[!n %in% "p43"], collapse = " + ")))

# La fórmula y~. no se acepta en la función neuralnet. Primero se debe escribir la fórmula
# y luego pasarla como un argumento en la función de ajuste.

# Se genera el modelo mediante la red neuronal.
# hidden se indica como un vector el cual contiene las capas ocultas en este caso
# la primera capa tiene 2 nodos y la segunda
# capa oculta tiene 3 (esto dependera del (estadisico:quien programe))

nn <- neuralnet(f,data=train_,hidden=c(2,3)) # aqui genero mi modelo para entrenar la red.

# Visual plot of the model (grafico de la red )

plot(nn)

summary(nn)

############ (prediccción de p43 utilizando la red neuronal) ########################

# Teniendo la red ya entrenada, podemos entregarle nuevas entradas no con el objetivo
# de que continúe aprendiendo,sino para obtener una predicción de cual debería ser el
# valor resultante de la función aprendida.Para ello usaremos la función compute().
# Para ello le facilitaremos la variable que contiene la configuración de la nn,
# un data.frame con los valores para las variables de entrada y, opcionalmente,
# indicaremos cuál de las repeticiones de la queremos usar.

pr.nn <- compute(nn,test_[,1:8])  # output=pr.nn

# Otra forma de hacer lo anterior.
prueb<-subset(test_,select=c("TIPOLOGI","INSTITUC","SEXO","edad","expdocente",
```

```
             "niveleduc","EXPINV","FORINV"))
prueb
pr.nn<-compute(nn,prueb)
pr.nn

# Results from NN are normalized (scaled)
# Descaling for comparison

# Ahora se puede predecir los valores para el conjunto de prueba y calcular el MSE.
# Recordar que la red dará salida a una predicción normalizada, por lo que tenemos
# que escalar de nuevo para hacer una comparación significativa
# (o simplemente una simple predicción).

pr.nn_ <- pr.nn$net.result*(max(datosfinal$p43)-min(datosfinal$p43))+min(datosfinal$p43)
test.r <- (test_$p43)*(max(datosfinal$p43)-min(datosfinal$p43))+min(datosfinal$p43)

# Calculating MSE
MSE.nn <- sum((test.r - pr.nn_)^2)/nrow(test_)
MSE.nn
#raiz cuadrada del error del modelo
SQRT_MSE<-sqrt(MSE.nn)
print(SQRT_MSE)

####################################

# Calcula el error cuadratico medio RMS.
yest<-nn$net.result
y<-data.frame(train$p43)
rms<-((sum((yest-y)^2)))/nrow(train)
print(rms)

# Raíz cuadrada del error del modelo.
SQRT_RMS<-sqrt(rms)
print(SQRT_RMS)

#############################################################################

# Predicción del modelo a partir de datos obtenidos del test (datos nuevos).

output <- compute(nn, test_[ ,c("TIPOLOGI","INSTITUC","SEXO","edad",
                       "expdocente","niveleduc","EXPINV","FORINV")])
output
#genera una tabla que muestra la respuesta real vs la obtenida por el modelo

result <- data.frame(
  Real = test$p43,
  Predicted =round(output$net.result,0),Error = abs(test$p43 - output$net.result) / test$p43 )
print(result)

# Calcula nuevamente el error a partir de las predicciones.

yest<-output$net.result
```

```
y<-data.frame(test$p43)
rms_test<-((sum((yest-y)^2)))/nrow(test)
print(rms_test)
#raiz cuadrada del error del modelo
print(sqrt(rms_test))


######################################################################

# Fitting linear model.
lm.fit <- glm(p43~., data=train)
summary(lm.fit)

# Predicted data from lm.
pr.lm <- predict(lm.fit,test)
pr.lm

# Test MSE.
MSE.lm <- sum((pr.lm - test$p43)^2)/nrow(test)
MSE.lm

# Compare the two MSEs.
print(paste(MSE.lm,MSE.nn))

# Plot predictions.

par(mfrow=c(1,2))

plot(test$p43,pr.nn_,col='red',main='Real vs predicted NN',pch=18,cex=0.7)
abline(0,1,lwd=2)
legend('bottomright',legend='NN',pch=18,col='red', bty='n')

plot(test$p43,pr.lm,col='blue',main='Real vs predicted lm',pch=18, cex=0.7)
abline(0,1,lwd=2)
legend('bottomright',legend='LM',pch=18,col='blue', bty='n', cex=.95)

# Compare predictions on the same plot.

plot(test$p43,pr.nn_,col='red',main='Real vs predicted NN',pch=18,cex=0.7)
points(test$p43,pr.lm,col='blue',pch=18,cex=0.7)
abline(0,1,lwd=2)
legend('bottomright',legend=c('NN','LM'),pch=18,col=c('red','blue'))


# Cross validating.

# Linear model cross validation
lm.fit <- glm(p43~.,data=datosfinal)
cv.glm(datosfinal,lm.fit,K=10)$delta[1] # esto no me sale

######################validacion cruzada red neuronal####################
```

```
# Neural net cross validation (validacion cruzada para la red neuronal)
set.seed(450)
cv.error <- NULL
k <- 10

# Initialize progress bar
library(plyr)
pbar <- create_progress_bar('text')
pbar$init(k)

i=1
for(i in 1:k){

    index <- sample(1:nrow(datosfinal),round(0.9*nrow(datosfinal)))
    train.cv <- scaled[index,]
    test.cv <-  scaled[-index,]
    # 1 NODOS PRIMERA CAPA OCULTA Y 1 LA SEGUNDA HACE HIDDEN.
    nn <- neuralnet(f,data=train.cv,hidden=c(1,1),linear.output=T)
    pr.nn <- compute(nn,test.cv[,1:8])
   # pr.nn<-compute(nn,prueb)

   # test.cv.r <- (test.cv$p43)*(max(datosfinal$p43)-min(datosfinal$p43))+min(datosfinal$p43)
     pr.nn <- pr.nn$net.result*(max(datosfinal$p43)-min(datosfinal$p43))+min(datosfinal$p43)
     test.cv.r <- (test.cv$p43)*(max(datosfinal$p43)-min(datosfinal$p43))+min(datosfinal$p43)

     #cv.error[i] <- sum((test.cv.r - pr.nn)^2)/nrow(test.cv)
      cv.error[i] <- sum((test.cv.r - pr.nn)^2)/nrow(test.cv)

     pbar$step()
}


# Average MSE ( ERROR CUADRATICO MEDIO)
mean(cv.error)

# MSE vector from CV
cv.error

# Visual plot of CV results
boxplot(cv.error,xlab='MSE CV',col='cyan',
        border='blue',names='CV error (MSE)',
        main='CV error (MSE) for NN',horizontal=TRUE)
```

### 12.1.2. Support Vector Machine.

```
######################### SVM   ########################
#leer datos desde excel
datos<-file.choose()
datosfinal<-read.csv2(datos,h=T)

#vista de datos en R
```

```
View(datosfinal)

#codificación
datosfinal$p43<-ifelse(datosfinal$p43==1 | datosfinal$p43==2 | datosfinal$p43==3,1,2 )
View(datosfinal$p43)

View(datosfinal) # Ahora me aparece codificada mi variable dependiente.
str(datosfinal)

# load the standard svm library (cargar paquete).
library(e1071)

# cargar e instalar paquete ggplot2.
library(ggplot2)
# se observa que es dificil de separar los datos
qplot(EXPINV,FORINV,data=datosfinal,color=p43)

# Otra forma de graficar.
plot(EXPINV~FORINV,data=datosfinal,col=as.factor(p43),
main="Grafico de Dispersión.")

# Support Vector Machine.Por defecto viene kernel radial en R
model<-svm(p43~.,data=datosfinal,type="C-classification")

plot(model,data=datosfinal,EXPINV~FORINV)
plot(model,data=datosfinal,TIPOLOGI~INSTITUC)
plot(model,data=datosfinal,SEXO~edad)
plot(model,data=datosfinal,expdocente~niveleduc)
plot(model,data=datosfinal,expdocente~niveleduc)

# Matriz de confución y error
pred<-predict(model,datosfinal)
tab<-table(predicted=pred,Actual=datosfinal$p43)
tab
# Error de predicción.
1-sum(diag(tab))/sum(tab)

################################################
 # Se toma una muestra de los datos.
 index<-sample(nrow(datosfinal), .9*nrow(datosfinal))
 train <- datosfinal[index,]     # datos para crear modelo.
 test <- datosfinal[-index,]     # datos para validar modelo.

#################### Support Vector Machine Lineal ########################
# Se estima (entrena) un modelo svm lineal para la muestra de entrenamiento
# y se predice la muestra de test.

modelo1<-svm(p43~.,data=train,kernel='linear',type="C-classification",
             scale=FALSE)
summary(modelo1)[7]

modelo1$index # vectores de soporte
```

```
# Coeficientes por los que se multiplican las observaciones para obtener
# el vector perpendicular al hiperplano que resuelve el problema.

modelo1$coefs

# Termino independiente.
modelo1$rho

plot(modelo1,data=train,EXPINV~FORINV)

# tune

obj1 <- tune.svm(p43~., data =train,kernel="linear", gamma = 2^(-1:1), cost = 2^(1:3))
summary(obj1)
plot(obj1)

best.mod1=obj1$best.model
summary(best.mod1)

# Otra forma para función tune.
set.seed(1)
tune.out=tune(svm,p43~.,data=train ,kernel="linear",
ranges=list(cost=c(0.001, 0.01, 0.1, 1,5,10,100) ))

summary(tune.out)

bestmod=tune.out$best.model
summary(bestmod)


# Predicción para la muestra test.
prediccion=predict(modelo1,newdata=test[,-9])
summary(prediccion)

# Matriz de confución.
mc<-table(test[,9],prediccion)
mc
t(mc)

aciertos=sum(diag(mc))/sum(mc)
aciertos

1-sum(diag(mc))/sum(mc)


######## VALIDACION CRUZADA  PARA MODELO CON KERNEL LINEAL ###########
Folds<-10
datosfinal$kfold<-sample(1:Folds, nrow(datosfinal),replace=T)
Iter<-data.frame(iteracion=NULL, aciertos=NULL)

i=1
```

```
for(i in 1:Folds)
{
    Test          <-subset(datosfinal, kfold==i)
    Entrenamiento <-subset(datosfinal,!kfold==i)

  # Training
  MODELO<-svm(p43~.,data=Entrenamiento,kernel='linear',type="C-classification")

  # Testing
  Prediccion    <-predict(MODELO, Test, type="class")
  MC            <-table(Test[,"p43"],Prediccion)
  Aciertos      <-MC[1,1]/(MC[1,1] + MC[2,1])
  Iter          <-rbind(Iter,data.frame(Iter=i,acierto=Aciertos))
}

# gráfico.
promedio <-format(mean(Iter$acierto, na.rm=TRUE)*100,digits=4)
plot(Iter,type="b",main="% prediccion en cada iteracion",
     cex.axis=1,cex.lab=1, cex.main=1,
     xlab="nro de iteraciones",ylab="% prediccion")
abline(h=mean(Iter$acierto),col="blue",lty=2)
legend("topright",legend=paste("eficiencia de prediccion=",promedio,"%"),
       col="blue", lty=1,lwd= 1, cex=1,bg=NULL)

#################### support vector machine radial ############################

modelo2<-svm(p43~.,data=train,kernel='radial',type="C-classification")
plot(modelo2,data=train,EXPINV~FORINV)

# Predicción para la muestra test.
prediccion2=predict(modelo2,newdata=test[,-9])
summary(prediccion2)

#Matriz de confusión.
mc2<-table(test[,9],prediccion2)
mc2

aciertos=sum(diag(mc2))/sum(mc2)
aciertos

1-sum(diag(mc2))/sum(mc2)

obj2 <- tune.svm(p43~., data =train,kernel="radial", gamma = 2^(-1:1), cost = 2^(1:3))
summary(obj2)[1]
plot(obj2)

best.param=obj2$best.parameters
summary(best.param)

###############  VALIDACION CRUZADA  PARA MODELO CON KERNEL RADIAL############
#k = 10 # Number of k-folds
Folds<-10
```

```
#id = sample(1:k,nrow(datosfinal),replace=TRUE)
datosfinal$kfold<-sample(1:Folds, nrow(datosfinal),replace=T)

#list = 1:k
Iter<-data.frame(iteracion=NULL, aciertos=NULL)

i=1
for(i in 1:Folds)
{
    Test          <-subset(datosfinal, kfold==i)
    Entrenamiento <-subset(datosfinal,!kfold==i)

  # Training
  MODELO<-svm(p43~.,data=Entrenamiento,kernel='radial',type="C-classification")


  # Testing
  #pred = predict(MODELOR, testset, type="response")
  Prediccion    <-predict(MODELO, Test, type="class")
  MC            <-table(Test[,"p43"],Prediccion)
  Aciertos      <-MC[1,1]/(MC[1,1] + MC[2,1])
  #real = Test$p43
  #rmse =  sqrt(sum((prediccion - real) ^ 2))/length(real)
  Iter          <-rbind(Iter,data.frame(Iter=i,acierto=Aciertos))
}

# gráfico.
promedio <-format(mean(Iter$acierto, na.rm=TRUE)*100,digits=4)
plot(Iter,type="b",main="% prediccion en cada iteracion",
     cex.axis=1,cex.lab=1, cex.main=1,
     xlab="nro de iteraciones",ylab="% prediccion")
abline(h=mean(Iter$acierto),col="blue",lty=2)
legend("topright",legend=paste("eficiencia de prediccion=",promedio,"%"),
       col="blue", lty=1,lwd= 1, cex=1,bg=NULL)

#################### Support Vector Machine Polinomial ####################

modelo3<-svm(p43~.,data=train,kernel="polynomial",type="C-classification")
plot(modelo3,data=train,EXPINV~FORINV)

# Predicción para la muestra test.
prediccion=predict(modelo3,newdata=test[,-9])
summary(prediccion)

#Matriz de confusión.
mc3<-table(test[,9],prediccion)
mc3
aciertos=sum(diag(mc3))/sum(mc3)
aciertos

1-sum(diag(mc3))/sum(mc3)
```

```
obj3 <- tune.svm(p43~., data =train,kernel="polynomial", gamma = 2^(-1:1), cost = 2^(1:3))
summary(obj3)[1]
plot(obj3)

best.param=obj3$best.parameters
summary(best.param)

########VALIDACION CRUZADA  PARA MODELO CON KERNEL POLINOMIAL ###########

Folds<-10
datosfinal$kfold<-sample(1:Folds, nrow(datosfinal),replace=T)
Iter<-data.frame(iteracion=NULL, aciertos=NULL)

i=1
for(i in 1:Folds)
{
    Test          <-subset(datosfinal, kfold==i)
    Entrenamiento <-subset(datosfinal,!kfold==i)

  # Training
  MODELO<-svm(p43~.,data=Entrenamiento,kernel='polynomial',type="C-classification")

  # Testing
  Prediccion    <-predict(MODELO, Test, type="class")
  MC            <-table(Test[,"p43"],Prediccion)
  Aciertos      <-MC[1,1]/(MC[1,1] + MC[2,1])
  Iter          <-rbind(Iter,data.frame(Iter=i,acierto=Aciertos))
}

# gráfico
promedio <-format(mean(Iter$acierto, na.rm=TRUE)*100,digits=4)
plot(Iter,type="b",main="% prediccion en cada iteracion",
     cex.axis=1,cex.lab=1, cex.main=1,
     xlab="nro de iteraciones",ylab="% prediccion")
abline(h=mean(Iter$acierto),col="blue",lty=2)
legend("topright",legend=paste("eficiencia de prediccion=",promedio,"%"),
       col="blue", lty=1,lwd= 1, cex=1,bg=NULL)

#################### support vector machine sigmoidal ############################

modelo4<-svm(p43~.,data=train,kernel="sigmoid",type="C-classification")
plot(modelo4,data=train,EXPINV~FORINV)

# Predicción para la muestra test , MATRIZ DE CONFUCION
prediccion=predict(modelo4,newdata=test[,-9])
summary(prediccion)
mc4<-table(test[,9],prediccion)
mc4

aciertos=sum(diag(mc4))/sum(mc4)
aciertos
```

```
1-sum(diag(mc4))/sum(mc4)

obj4 <- tune.svm(p43~., data =train,kernel="sigmoid", gamma = 2^(-1:1), cost = 2^(1:3))
summary(obj4)
plot(obj4)

best.param=obj4$best.parameters
summary(best.param)

########VALIDACION CRUZADA  PARA MODELO CON KERNEL SIGMOIDAL ############
Folds<-10
datosfinal$kfold<-sample(1:Folds, nrow(datosfinal),replace=T)
Iter<-data.frame(iteracion=NULL, aciertos=NULL)

i=1
for(i in 1:Folds)
{
    Test          <-subset(datosfinal, kfold==i)
    Entrenamiento <-subset(datosfinal,!kfold==i)

  # Training
  MODELO<-svm(p43~.,data=Entrenamiento,kernel='sigmoid',type="C-classification")

  # Testing
  Prediccion    <-predict(MODELO, Test, type="class")
  MC            <-table(Test[,"p43"],Prediccion)
  Aciertos      <-MC[1,1]/(MC[1,1] + MC[2,1])
  Iter          <-rbind(Iter,data.frame(Iter=i,acierto=Aciertos))
}

# gráfico
promedio <-format(mean(Iter$acierto, na.rm=TRUE)*100,digits=4)
plot(Iter,type="b",main="% prediccion en cada iteracion",
     cex.axis=1,cex.lab=1, cex.main=1,
     xlab="nro de iteraciones",ylab="% prediccion")
abline(h=mean(Iter$acierto),col="blue",lty=2)
legend("topright",legend=paste("eficiencia de prediccion=",promedio,"%"),
       col="blue", lty=1,lwd= 1, cex=1,bg=NULL)

#################### Support Vector Machine Gausiano ############################

library("kernlab")

modelgauss <- ksvm(p43 ~ .,data=train ,type = "C-bsvc", kernel = "rbfdot",
             kpar = list(sigma = 0.1), C = 1,
             prob.model = TRUE)

print(modelgauss )

#plot(modelgauss,data=train,EXPINV~FORINV)

# Predicción para la muestra test , MATRIZ DE CONFUCION
```

```
predmodelgauss=predict(modelgauss,newdata=test[,-9])
summary(predmodelgauss)
mc5<-table(test[,9],predmodelgauss)
mc5

aciertos=sum(diag(mc5))/sum(mc5)
aciertos

1-sum(diag(mc5))/sum(mc5)

obj5 <- tune.svm(p43~., data =train,type = "C-bsvc", kernel = "rbfdot",
             kpar = list(sigma = 0.1), C = 1,
             prob.model = TRUE)

summary(obj4)
plot(obj4)

best.param=obj4$best.parameters
summary(best.param)
#########VALIDACION CRUZADA  PARA MODELO CON KERNEL GAUSIANO ############
Folds<-10
datosfinal$kfold<-sample(1:Folds, nrow(datosfinal),replace=T)
Iter<-data.frame(iteracion=NULL, aciertos=NULL)

i=1
for(i in 1:Folds)
{
    Test          <-subset(datosfinal, kfold==i)
    Entrenamiento <-subset(datosfinal,!kfold==i)

  # Training
  library("kernlab")
  MODELO<- ksvm(p43 ~ .,data=Entrenamiento ,type = "C-bsvc", kernel = "rbfdot",
             kpar = list(sigma = 0.1), C = 1,
             prob.model = TRUE)

  # Testing
  #Prediccion    <-predict(MODELO, Test, type="class")
   Prediccion =predict(MODELO,Test)
  MC            <-table(Test[,"p43"],Prediccion)
  Aciertos      <-MC[1,1]/(MC[1,1] + MC[2,1])
  Iter          <-rbind(Iter,data.frame(Iter=i,acierto=Aciertos))
}

# grafico
promedio <-format(mean(Iter$acierto, na.rm=TRUE)*100,digits=4)
plot(Iter,type="b",main="% prediccion en cada iteracion",
     cex.axis=1,cex.lab=1, cex.main=1,
     xlab="nro de iteraciones",ylab="% prediccion")
abline(h=mean(Iter$acierto),col="blue",lty=2)
legend("topright",legend=paste("eficiencia de prediccion=",promedio,"%"),
       col="blue", lty=1,lwd= 1, cex=1,bg=NULL)
```

```
################################################################

SVMLINEAL<- c(0.7000000, 0.6666667, 0.6000000, 0.6000000, 0.5000000,
              0.6250000, 0.7142857, 0.5454545, 0.2727273, 1.0000000)

SVMRADIAL<- c(0.5333333, 1.0000000, 0.6666667, 0.7857143, 0.5714286,
              0.4705882, 0.3571429, 0.6470588, 0.7142857, 0.4705882)

SVMPOLY<- c(0.5384615, 0.5333333, 0.6818182, 0.4166667, 0.3333333,
            0.6250000, 0.5384615, 0.6000000, 0.4285714, 0.5555556)

SVMSIGMO<- c(0.7777778, 0.5454545, 0.2727273, 0.7000000, 0.5714286,
             0.6666667, 0.8000000, 0.6666667, 0.5500000, 0.6250000)

SVMGAUSS<- c(0.6250000, 0.5000000, 0.6666667, 0.6363636, 0.6923077,
             0.8000000, 0.4666667, 0.3750000, 0.5454545, 0.4545455)


#plot(x,type="l")#(tipos= l,o,p,b,h,s,c)
#cambiar x,index y poner titulo e intervalo
# xaxp(comienza en 1,hasta 10 con 9 intervlos.
# ann me quita el nombre de titulo y de los ejes.
# con main="" quito el nombre del titulo.
# lwd=2 cambia el grosor de la linea
# axes=F quita el eje de cordenadas
# yaxt="n" quita los valores de ordenadas.
# xaxt="n"  quita los valores de las abscisas

# Haciendo el gráfico 1
plot(SVMLINEAL,              # Valores a graficar
     type="o",               # pinta líneas en el gráfico
     col="red",              # color rojo para la línea
     bty='l',                # para no pintar línea superior en la 'caja' del gráfico
     pch=1,
     lwd = 4,
     main="% prediccion en cada iteracion",  # Título
     ylab='% prediccion', xlab='nro de iteraciones',   #  etiquetas en los ejes
     las=1,                  # números de los ejes correctamente girados
     lty=1,
     cex.axis=.75 )          # tamaño de los números de los ejes

#lines(SVMLINEAL,lty=1)

# Haciendo el segundo gráfico y agregándoselo al anterior.
par(new=TRUE)                # permite sobreimponer un gráfico al anterior

plot(SVMRADIAL,
     type="b",
     col="blue",
     pch=2,
     lwd = 4,
```

```
     bty='n',                # n evita superponer líneas en la 'caja' del gráfico.
     xaxt="n",               # sin números el eje x, esto se hará luego con 'axis'
     yaxt="n",               # sin números el eje y, esto se hará luego con 'axis'
     xlab="", ylab="",
     lty=2,
     cex.axis=.75)

#lines(SVMRADIAL,lty=2)

# Haciendo el tercer gráfico y agregándoselo al anterior.
par(new=TRUE)

plot(SVMPOLY,
     type="l",
     col="green",
     pch=3,
     lwd=4,
     bty='n',
     xaxt="n",
     yaxt="n",
     xlab="", ylab="",
     lty=3,
     cex.axis=.75)

#lines(SVMPOLY,lty=3)

# Haciendo el cuarto gráfico y agregándoselo al anterior.
par(new=TRUE)                # permite sobreimponer un gráfico al anterior

plot(SVMSIGMO,
     type="o",
     col="black",
     pch=4,
     lwd =4,
     bty='n',
     xaxt="n",
     yaxt="n",
     xlab="", ylab="",
     lty=4,
     cex.axis=.75)

#lines(SVMSIGMO,lty=4)

# Haciendo el quinto gráfico y agregándoselo al anterior.
par(new=TRUE)

plot(SVMGAUSS,
     type="l",
     col="orange",
     pch=5,
     lwd =4,
     bty='n',
```

```
     xaxt="n",
     yaxt="n",
     xlab="", ylab="",
     lty=5,
     cex.axis=.75)

#lines(SVMGAUSS,lty=5)

legend("topright",    # ubicación de la leyenda 'parte superior derecha'
  # contenido de la leyenda
  legend=c("SVMLINEAL","SVMRADIAL","SVMPOLY","SVMSIGMO","SVMGAUSS"),
  col=c("red","blue","green","black","orange"),  # colores de líneas
  lty=1:5,              # tipo de línea: línea sólida
  lwd = 4,              # grosor de la linea
  cex=0.8,              # tamaño del contenido de la leyenda
  bty='n')              # sin cuadro que rodee a la leyenda.


############   PROGRAMACION MEJOR KERNEL Y COSTO      ########

C<-10
svm.lineal<- svm(p43~., data=train, kernel='linear',cost=C,type="C-classification",
                 cross=2, scale=FALSE)
svm.lineal
summary(svm.lineal)[29]

names(svm.lineal)[29]

#########################################################

Kernel<-c("linear","polynomial","radial","sigmoid")
C<-c(10,50,100,500,1000)
P<-matrix(0,length(Kernel),length(C))
rownames(P)=Kernel<-c("linear","polynomial","radial","sigmoid")#NOMBRE FILAS
colnames(P)=C<-c(10,50,100,500,1000)         #NOMBRE COLUMNAS
P
i=1

for(i in 1:length(Kernel)) {
    t<-0
    for(j in C) {
      t<-t+1
      svm.lineal <- svm(p43~., data=train, kernel=Kernel[i],cost=j, cross=2,scale=FALSE,
                        type="C-classification")
      P[i,t]<-summary(svm.lineal)$tot.accuracy
   }
}
P

max(P)
min(P)
P[1,1]
```

```
P[1,2]
dim(P)
dimnames(P)

# mejor kernel y costo
which(P==max(P),arr.ind = TRUE)

# peor escenario
which(P==min(P),arr.ind = TRUE)
```