Aprendizaje Automático (2014-2015)
Grado en Ingeniería Informática
Universidad de Granada

Práctica 3

Ignacio Martín Requena

31 de mayo de 2015

Índice

Índice de figuras

1. Cuestionario

1.1. Suponga que dispone de una muestra i.i.d para estudiar la predicción del valor de una variable Y para un valor dado del predictor X. Suponga que elige al azar uno de los métodos estudiados. ¿Cómo podríamos estimar la desviación típica de nuestra predicción? Dar todos los detalles de cada paso.

Cogeriamos un subconjunto de datos de training y construiriamos el modelo a partir de este, a continuación calcularíamos la media que el modelo ha generado pasandole el conjunto de test y obteniendo los resultados del mismo. Una vez obtenida la media aritmética calcularíamos la desviación típica con la fórmula:

1.2. Describir que problema resuelve y cuál es el fundamento de la técnica de Validación Cruzada de k-partes (k-CV) y porque debe de funcionar.

Al usar validación cruzada evitas que la muestra de train tenga datos "especiales" y obtengamos un modelo poco ajustado con la realidad.
Al hacer validación cruzada, de alguna manera cogemos todo el conjunto de muestras como train menos una porción del mismo que la usaremos como test y asi con tantas conbinaciones como especifique el valor de K.
Es decir, resuelve el problema de que cojamos un conjunto con muestras poco representativas.

1.3. Describir las ventajas y desventajas de usar k-CV respecto de usar una aproximación basada en un conjunto de validación o en Leave-One-Out (LOO).

- **Ventajas KCV**: El tiempo de computo con respecto a LOO es menor, asegura que test y training estarán mas equilibrados y obtenemos mejor error de test respecto a los otros modelos.
- **Desventajas KCV**: LOO hace que el modelo aprenda de mas muestras que en KCV, KCV es mas difícil de implementar frente a un solo conjunto de validación, hay un error de test variable porque pueden caer muestras poco representativas.

1.4. ¿En que beneficia la combinación de múltiples clasificadores frente al uso de un único clasificador? Justificar la respuesta

Al usar multiples clasificadores obtenemos una mayor capacidad de contraste debido a la mayor informacion que poseemos sobre que clasificadores funcionan mejor sobre el modelo. A demás el uso de multiples clasificadores hace que tengamos mas liberta de modelizacion.

1.5. ¿Qué es y que aporta el predictor Random Forest frente al uso de Bagging con árboles? Justificar la respuesta.

Random Forest esta construido sobre la ídea de bagging pero además también aporta una mejora ya que construye árboles no correlados.

Si existe un predictor muy fuerte en el conjunto de datos entonces muchos de ellos usarán dicho predictor para la primera partición.

El promedio de cantidades altamente correladas no reduce mucho su varianza y por tanto "random forest" correla los árboles de bagging para obtener una mayor reducción en varianza.

1.6. Si tenemos dos métodos que son capaces de separar linealmente un problema de dos clases y uno de ellos es SVM-lineal. ¿Hay alguna razón que nos llevarían a preferir la técnica SVM frente al otro método? Justificar la respuesta

Si hay una clara diferenciación entre los datos de una clase y de otra sería mejor tomar el metodo SVM-lineal, ya que no se "sobreajusta" tanto a las muestras como el modelo plenamente lineal, si no que traza una frontera minima y, de alguna manera, consigue encontrar la zona bajo la cual las muestras se dividen.

1.7. ¿Cuáles son las razones principales para usar técnicas de núcleo en un problema dado? Describir los casos y justificar la respuesta.

Las tecnicas de núcleo son buenas para tener libertad a la hora de ajustar el modelo. Por ejemplo el Kernel radial nos da la posibilidad de definir un modelo a partir de la definición de un radio a partir del cual se situa la frontera de clasificación. Estas técnicas yo las usaria para problemas que necesiten un modelo algo diferente al resto.

1.8. En un laboratorio de biológicos se procesan muestras de material genético para obtener un modelo de predicción de cáncer. Debido al coste de procesamiento solo se pueden procesar un bajo número de muestras, sin embargo cada muestra proporciona un vector de variables de considerable longitud. Los investigadores son capaces de identificar que variables son relevantes como predictores y cuales como predicción, pero no saben que técnica sería más conveniente aplicar en este caso. Discutir el problema y proponer y justificar soluciones adecuadas desde el punto de vista metodológico

Eligiría un modelo que realizara pocas operaciones pero que fuera efectivo, ya que al tener tantisimas variables, un modelo como SVM podría tardar mucho en realizarse.

Un buen modelo podría ser validación cruzada o regresión logística. Validación cruzada con un modelo de clasificación por refresión logística nos daría buenos resultados sobre este conjunto de datos.

2. Ejercicios de implementacion

2.1. Usar el conjunto de datos OJ que es parte del paquete ISLR

2.1.1. Crear un conjunto de entrenamiento conteniendo una muestra aleatoria de 800 observaciones, y un conjunto de test conteniendo el resto de las observaciones. Ajustar un clasificador SVM (con núcleo lineal) a los datos de entrenamiento usando cost=0.01, con "Purchase" como la respuesta y las otras variables como predictores.

La salida que R nos muestra una vez ajustado el clasificador SVM es la siguiente:

```
Call:
svm(formula = Purchase ~ ., data = train, kernel = "linear", cost = 0.01,
    scale = FALSE)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  0.01
      gamma:  0.05555556

Number of Support Vectors:  615
```

Figura 2.1: Salida R SVN nucleo lineal

Como podemos ver R nos especifica que es de tipo clasificacion y con un nucleo lineal, que tiene un costo de 0.01 (por lo que al ser pequeño los margenes seran anchos y muchos vectores estaran en el margen) y un margen de hiperplanos de 0.0555

2.1.2. Usar la función summary() para producir un resumen estadístico, y describir los resultados obtenidos. ¿Cuáles son las tasas de error de "training" y "test"?

```
svm(formula = Purchase ~ ., data = train, kernel = "linear",
    cost = 0.01, scale = FALSE)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  0.01
      gamma:  0.05555556

Number of Support Vectors:  615

 ( 306 309 )


Number of Classes:  2

Levels:
 CH MM
```

Figura 2.2: Salida R SVN nucleo lineal (summary)

Además de lo comentado en el apartado anterior la funcion summary también nos muestra el número de support vectors y como están divididos en el modelo de clasificación, el numero de clases y los niveles.

En cuanto a la distribucion de los vectores se puede ver que es equilibrada (306 309), por lo que es posible que nuestro modelo sea bueno.
La tasa de error de train es del 26,25 % y la de test de 26,3 %, unos errores que aun pudiendo estar mas bajos no son del todo malos.

2.1.3. Usar las función tune() para seleccionar un coste óptimo. Considerar los valores de "cost" del vector: [0.001, 0.01, 0.1, 1, 10]. Dibujar las curvas ROC para los diferentes valores del "cost".

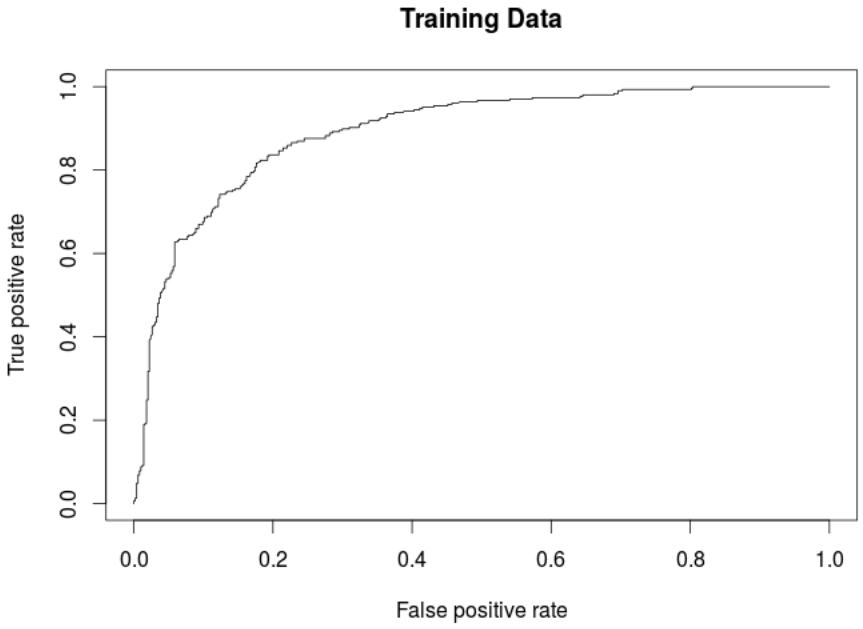

Figura 2.3: Curva ROC para cost = 0.001

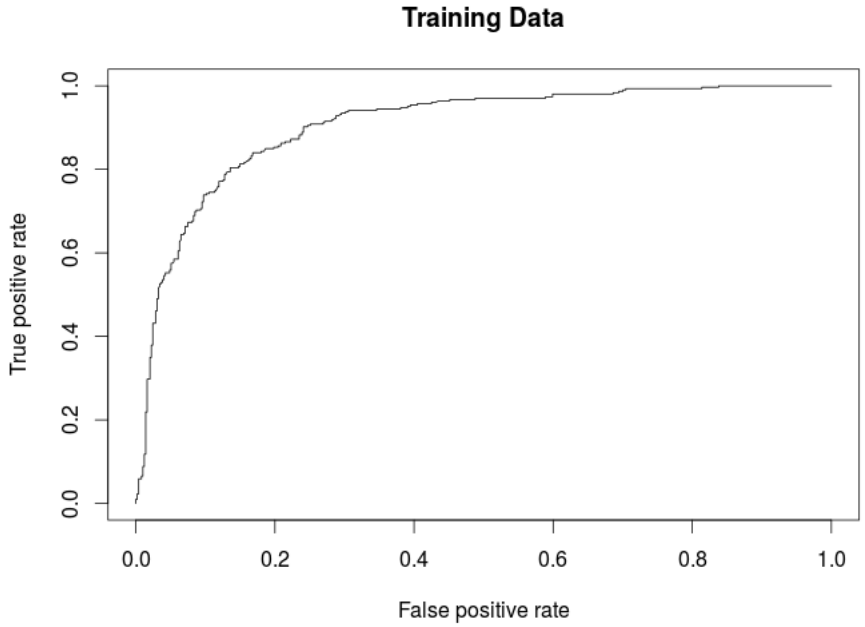

Figura 2.4: Curva ROC para cost = 0.01

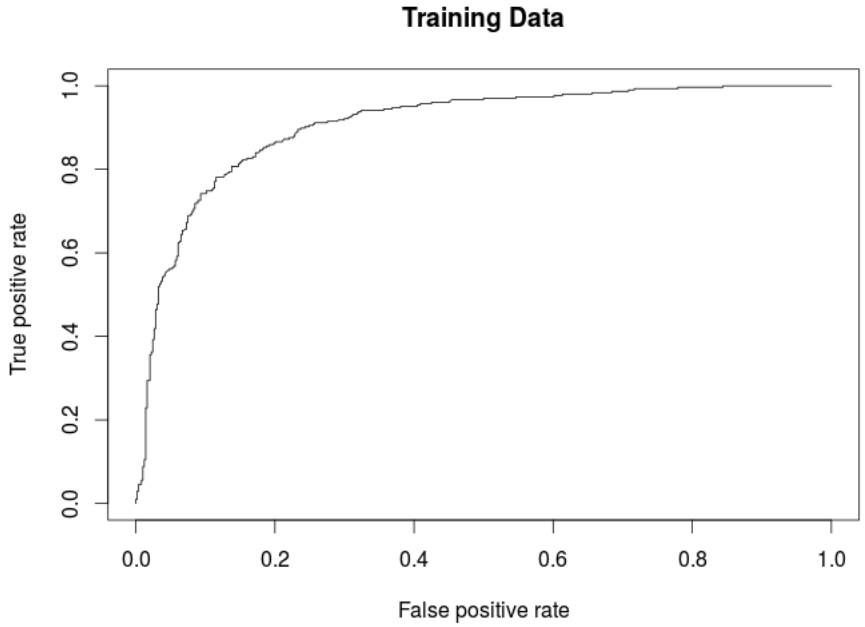

Figura 2.5: Curva ROC para cost = 0.1

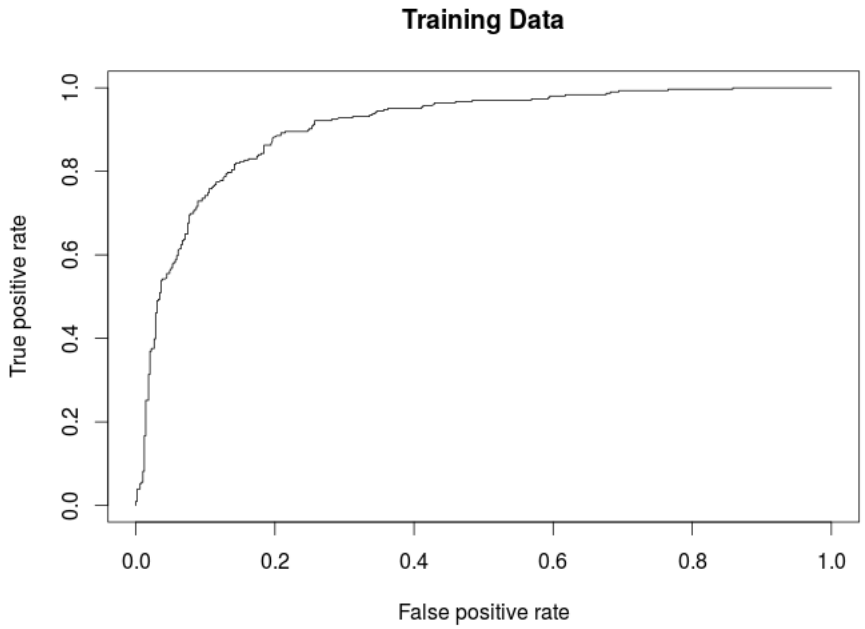

Figura 2.6: Curva ROC para cost = 1

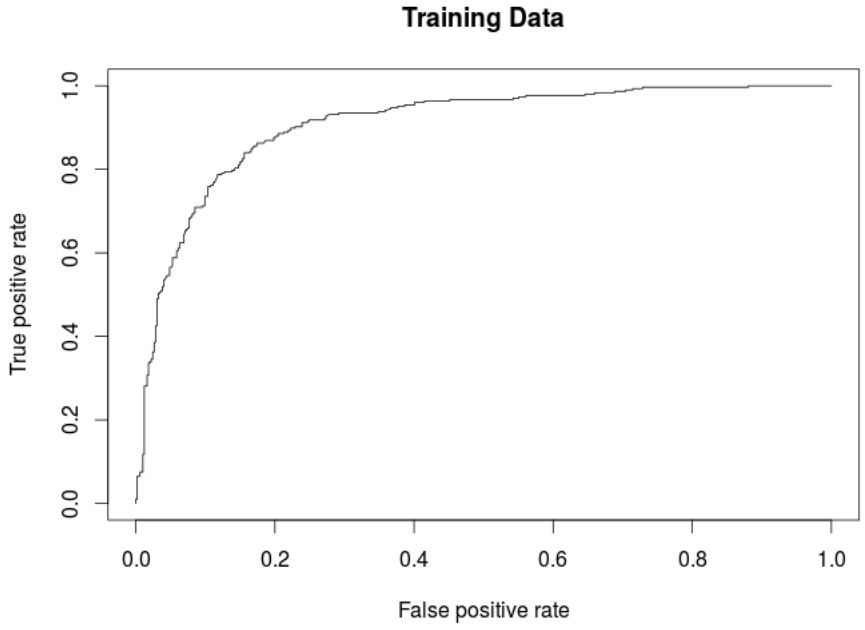

Figura 2.7: Curva ROC para cost = 10

El valor de coste óptimo del parámetro cost es de 0.01, con un rendimiento de 0.175. Como vemos en las curvas anteriores, si las comparamos con respecto a la calculada para el caso óptimo (cost = 0.01) vemos como esta es más continua (lo que hace que no haya mucha fluctuacion entre los ratios de verdaderos positivos) y por tanto nuestro modelo no nos mostrara valores raros ni saltos de probabiliades, para valores de cost altos la curva crece de forma más "exponencial".

2.1.4. Calcular las tasas de error de "training" y "test" usando el nuevo valor de coste óptimo.

Ahora nuestros errores han bajado hasta una tasa de error en train del 16,62 % y de test de 18,14 %, por lo que hemos comprobado que el nuevo valor de cost reduce aun mas la probabilidad de error con el nuevo valor óptimo de cost.

2.1.5. Repetir apartados (2) a (4) usando un SVM con núcleo radial. Usar valores de gamma en el rango [10, 1, 0.1, 0.01, 0.001]. Discutir los resultados

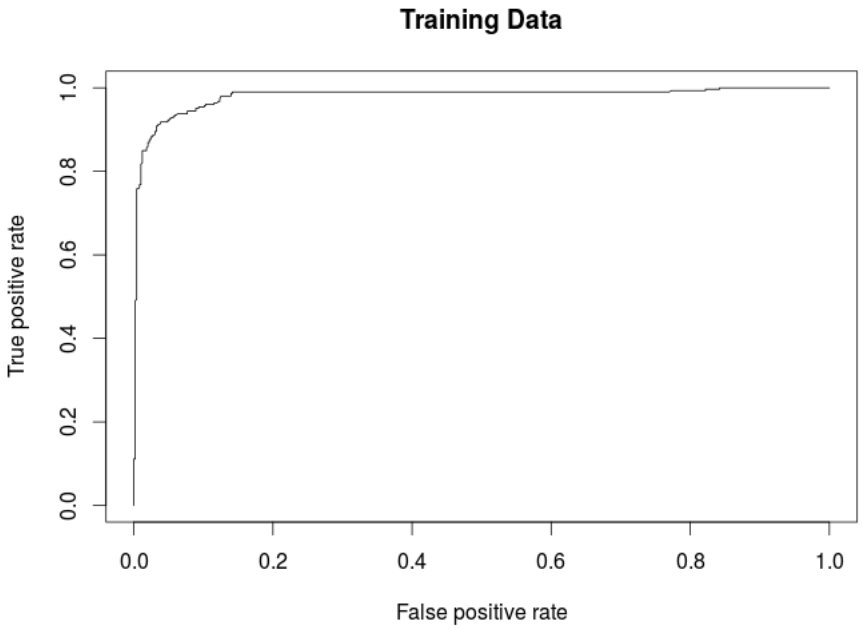

Figura 2.8: Curva ROC para gamma = 10

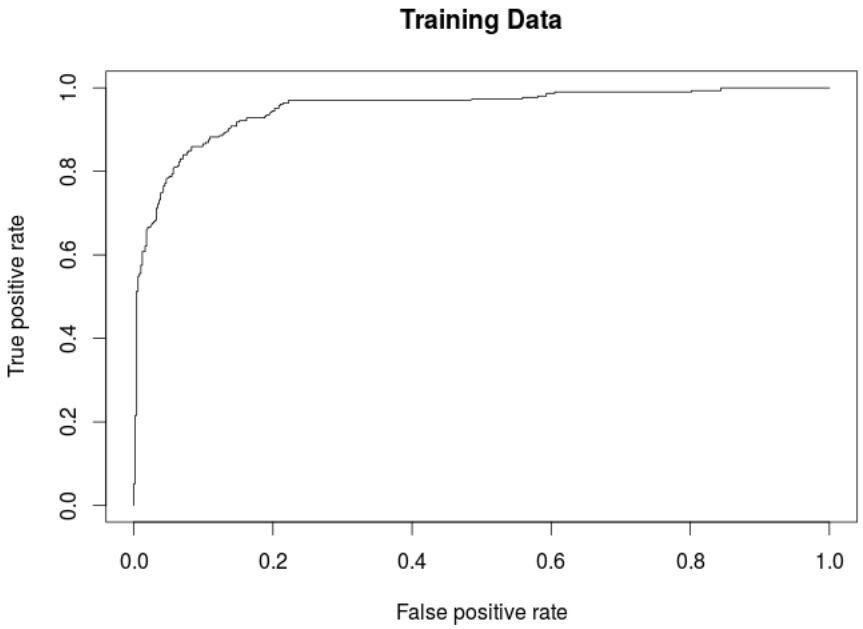

Figura 2.9: Curva ROC para gamma = 1

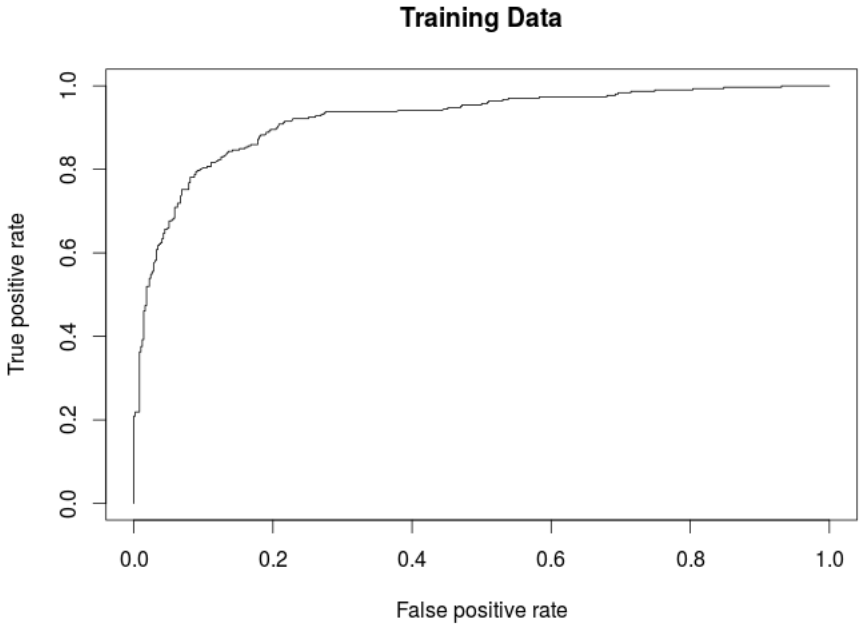

Figura 2.10: Curva ROC para gamma = 0.1

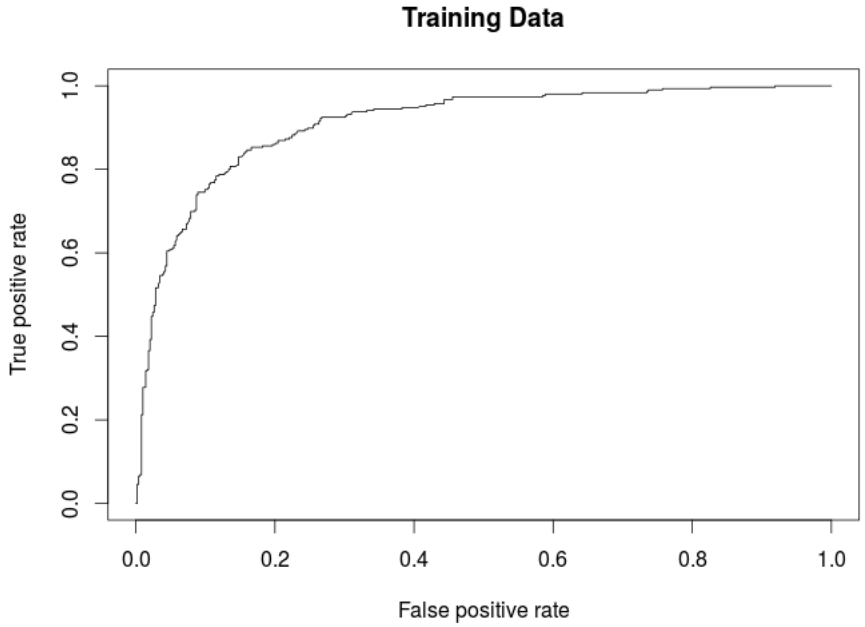

Figura 2.11: Curva ROC para gamma = 0.01

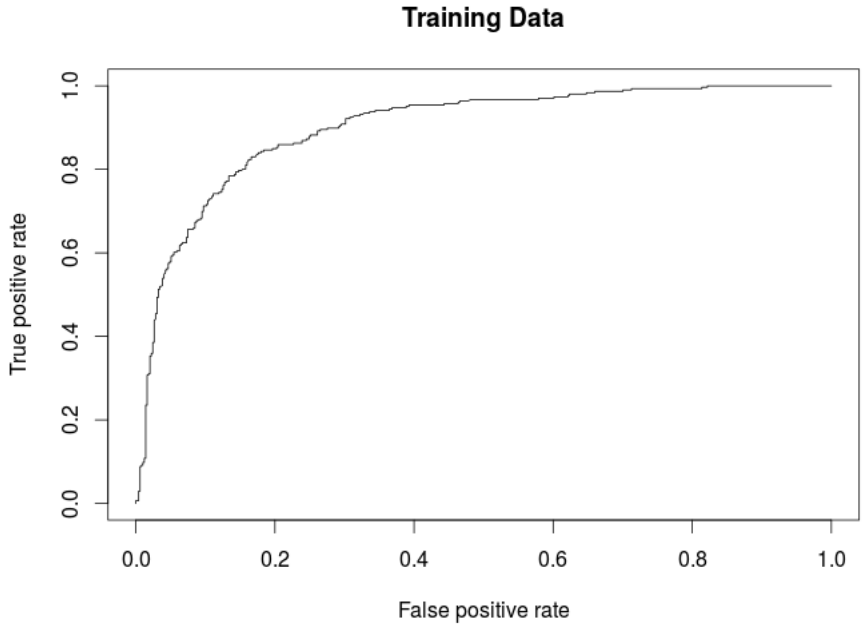

Figura 2.12: Curva ROC para gamma = 0.001

Al principio, con un gamma de valor 10 obtenemos unas tasas de error para training y test de 0.061 y 0.2370 respectivamente. Como podemos ver aunque el error no es muy desorvitado (el de test quiza si, depende de la incertidumbre que asumamos) estan muy descompensados, lo que nos hace inuir que no es un buen valor para el parametro.
Al calcular el valor optimo de gamma R nos dice que es 0.01, por lo que al ajustar el SVM con nucleo radial los errores de training y test se hacen mas proporcionados y por tanto mejores (0.1662 y 0.1814).
Por las razones comentadas en el ejercicio anterior, al comparar las curvas vemos como la que tiene un valor de 0.01 es mejor que el resto.

2.1.6. Repetir apartados (2) a (4) usando un SVM con un núcleo polinómico. Usar degree con valores 2,3,4,5,6. Discutir los resultados

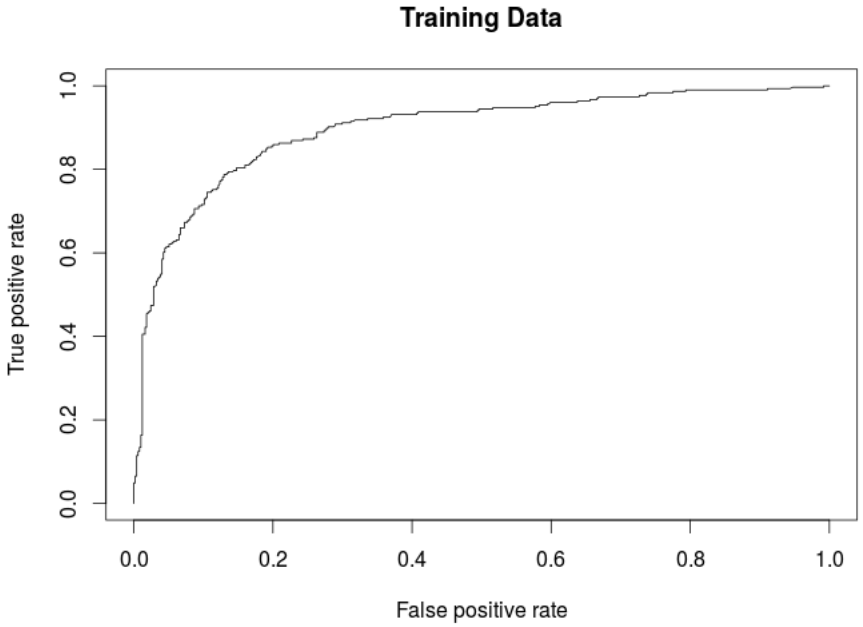

Figura 2.13: Curva ROC para degree = 0.001

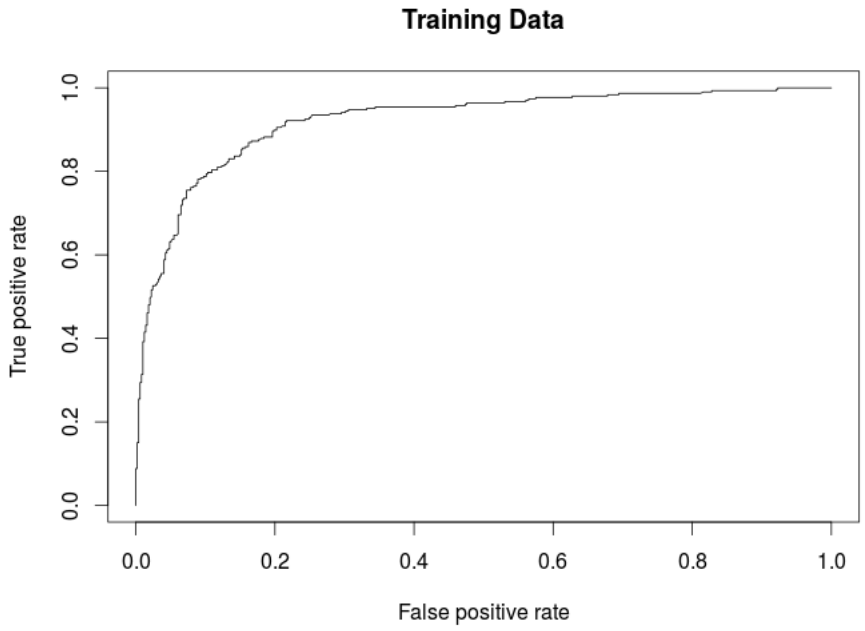

Figura 2.14: Curva ROC para degree = 0.001

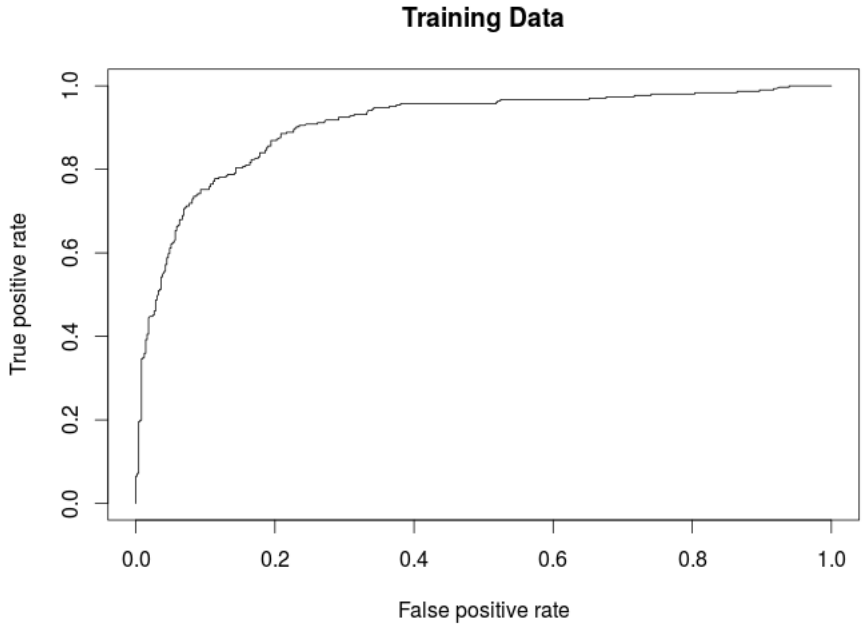

Figura 2.15: Curva ROC para degree = 0.001

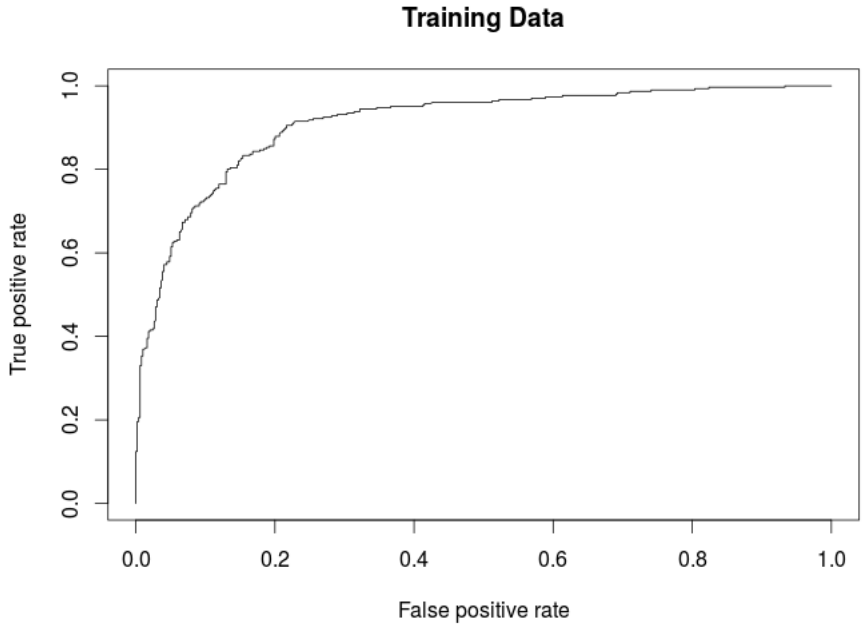

Figura 2.16: Curva ROC para degree = 0.001

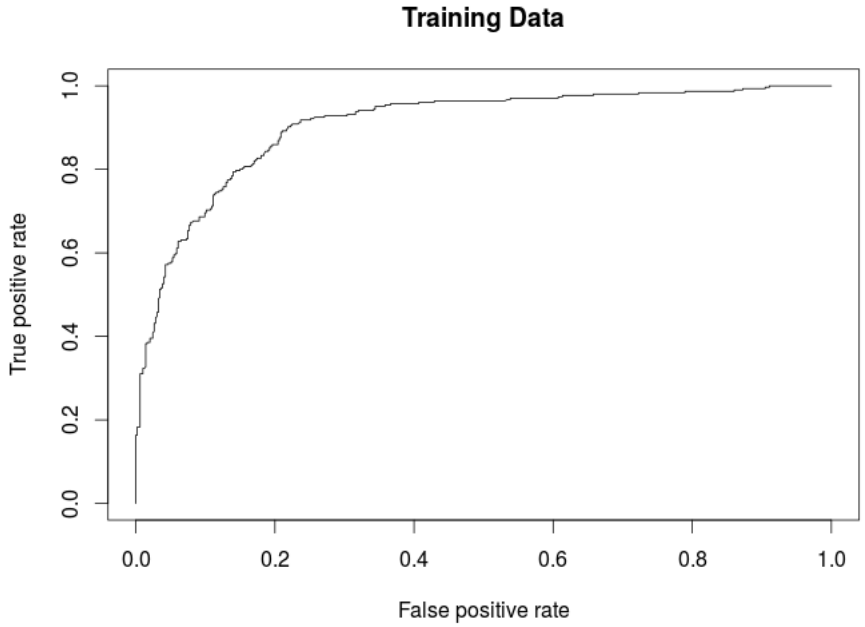

Figura 2.17: Curva ROC para degree = 0.001

Calculado el error de train y test para el primer valor de degree (degree = 2) el modelo nos da buenos valores (0.172 y 0.1888) por lo que o hemos acertado de casualidad a la hora de elegir el valor de degree o el modelo se ajusta mejor a los datos (sin llegar a sobreajustar).
Al calcular el valor óptimo de la variable degree este nos da 3. Nos habiamos aproximado al valor óptimo en el primer calculo pero aun asi se podia acercar mucho mas. Esto ya nos empieza a decir que quiza sea un buen modelo.
Los errores de train y test para degree = 2 son del 14,5 % y 17,03 %. No son valores muy lejanos y a demás son asumibles esos porcentajes de error.

2.1.7. En global, ¿qué aproximación da el mejor resultado sobre estos datos?

Teniendo en cuenta la siguiente tabla:

Kernels prámetros opt.	**% Error Train**	**% Error test**
Lineal	16,62	18,14
Radial	16,62	18,14
Polinómico	14,5	17,03

Como podemos ver los errores de train y test para los núcleos lineal y radial son los mismos, lo que nos indica que los datos se ajustan de igual manera a ambos nucleos.

Además el mejor modelo para usar como clasificador es el polinómico, ya que los errores de train y test bajan sin diferenciarse mucho uno de otro.

2.2. Usar el conjunto de datos OJ que es parte del paquete ISLR

2.2.1. Crear un conjunto de entrenamiento conteniendo una muestra aleatoria de 800 observaciones, y un conjunto de test conteniendo el resto de las observaciones. Ajustar un árbol a los datos de "training", usando "Purchase" como la respuesta y las otras variables excepto "Buy" como predictores

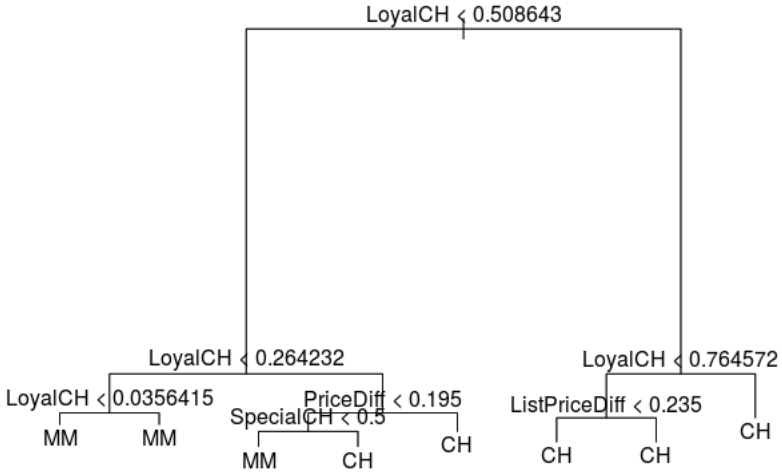

Figura 2.18: Árbol de datos de training

2.2.2. Usar la función summary() para generar un resumen estadístico acerca del árbol y describir los resultados obtenidos: tasa de error de "training", número de nodos del árbol, etc. Teclee el nombre del objeto árbol y obtendrá una salida en texto. Elija un nodo e interprete su contenido.

```
Classification tree:
tree(formula = Purchase ~ ., data = train)
Variables actually used in tree construction:
[1] "LoyalCH"       "PriceDiff"     "SpecialCH"     "ListPriceDiff"
Number of terminal nodes:  8
Residual mean deviance:  0.7305 = 578.6 / 792
Misclassification error rate: 0.165 = 132 / 800
```

Figura 2.19: Árbol de datos de training (summary)

La función summary nos muestra las variables que se han usado en el arbol, los nodos que este tiene, la desviacion media y el ratio de error de clasificacion.

El arbol no es muy extenso, lo que nos indica que el modelo tiene una complejidad aceptable (solamente 8 nodos). La desviacion media tiene un valor aceptable (0,7305) y un error también asumible (0,165)
Para el nodo 3 por ejemplo R nos muestra la siguiente información:

La salida en texto nos da la siguiente información:

node), split, n, deviance, yval, (yprob)

3) LoyalCH >0.508643 450 318.10 CH (0.88667 0.11333)

El nodo seleccionado es el nodo 3. Si la variable LoyalCH es mayor que 0.5086... seleccionaremos ese nodo, para expandir, tiene 450 muestras, una desviacion de 318.10 y un valor de y de CH con probabilidad entre 0.8866 y 0.1133.

2.2.3. Crear un dibujo del árbol. Extraiga las reglas de clasificación más relevantes definidas por el árbol (al menos 4).

El dibujo del arbol se muestra en el paartado 1 de esta cuestion. En cuanto a las reglas de clasificación mas relevantes:

1) root 800 1064.00 CH (0.6175 0.3825)
2) LoyalCH <0.508643 350 409.30 MM (0.2714 0.7286)
4) LoyalCH <0.264232 166 122.10 MM (0.1205 0.8795) *
5) LoyalCH >0.264232 184 248.80 MM (0.4076 0.5924)
10) PriceDiff <0.195 83 91.66 MM (0.2410 0.7590)
20) SpecialCH <0.5 70 60.89 MM (0.1571 0.8429) *
21) SpecialCH >0.5 13 16.05 CH (0.6923 0.3077) *
11) PriceDiff >0.195 101 139.20 CH (0.5446 0.4554) *
3) LoyalCH >0.508643 450 318.10 CH (0.8867 0.1133) *

Estas reglas nos las proporciona R con la función "misclass". Esta funcion determina una secuencia de subárboles del árbol recursivamente çortando.[en] las divisiones poco importantes.

2.2.4. Predecir la respuesta de los datos de test, y generar e interpretar la matriz de confusión de los datos de test. ¿Cuál es la tasa de error del test? ¿Cuál es la precisión del test?)

La matriz de confusión para los datos de test nos muestra que el porcentaje de acierto es medianamente elevado con respecto al conjunto de test (0.7740741). Por otro lado el error de test es de 22,29 %, un poco elevado pero asumible (un 77,40 % de acierto).

2.2.5. Aplicar la función cv.tree() al conjunto de "training" para determinar el tamaño óptimo del árbol.

El tamaño optimo del arbol 4, ya que segun nos indica la salida de cv.tree es àra el que se produce menor desviación.

```
$size
[1] 8 7 6 5 4 3 2 1

$dev
[1]  704.7613  703.8106  684.8658  686.8563  684.0398  732.5495
[7]  735.3661 1071.0002

$k
[1]      -Inf  11.20965  14.72877  17.88334  23.55203  38.37537
[7]  43.02529 337.08200

$method
[1] "deviance"

attr(,"class")
[1] "prune"         "tree.sequence"
```

Figura 2.20: Salida de cv.tree para el conjunto de training

2.2.6. Generar un gráfico con el tamaño del árbol en el eje x y la tasa de error de validación cruzada en el eje y. ¿Qué tamaño de árbol corresponde a la tasa más pequeña de error de clasificación por validación cruzada?

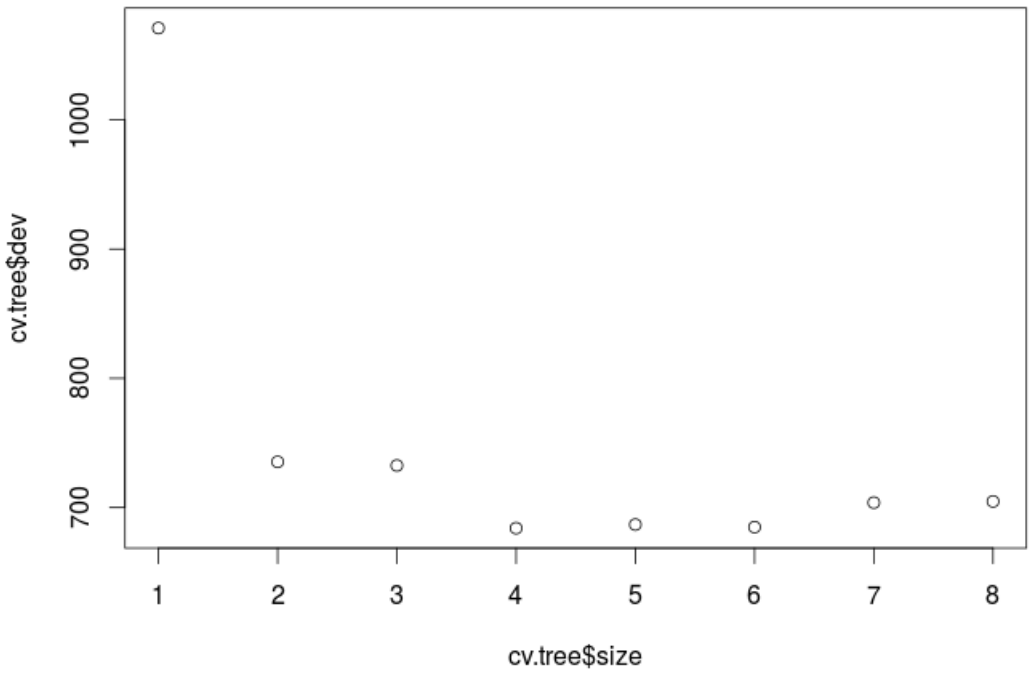

Figura 2.21: Gráfico tamaño árbol vs tasa de error VC

El tamaño del árbol para la tasa mas pequeña de validacion cruzada es el de tamaño 4. Supongo que sera porque en este tamaño habra un compromiso entre tamaño y cantidad de informacion necesaria para que sea un buen modelo.

2.2.7. Ajustar el árbol podado correspondiente al valor óptimo obtenido en 6. Comparar los errores sobre el conjunto de training y test de los árboles ajustados en 6 con el árbol podado. ¿Cuál es mayor?

La siguiente tabla representa la comparacion entre los errores de train y test para el árbol podado y sin podar:

Errores	**Sin podar**	**Podado**
Train	0.165	0.165
Test	0.2259	0.2259

Como podemos ver tanto el erro de train como de test con el arbol podado y sin podar

es el mismo, lo que nos indica que en este caso la poda del arbol para el valor óptimo obtenido no es necesaria.

2.3. Sobre el conjunto de datos Hitters

2.3.1. Realizar boosting sobre el conjunto de entrenamiento con 1,000 árboles para un rango de valores del parámetro de ponderación lambda. Realizar un gráfico con el eje x mostrando diferentes valores de lambda y los correspondientes valores de MSE de "training" sobre el eje y.

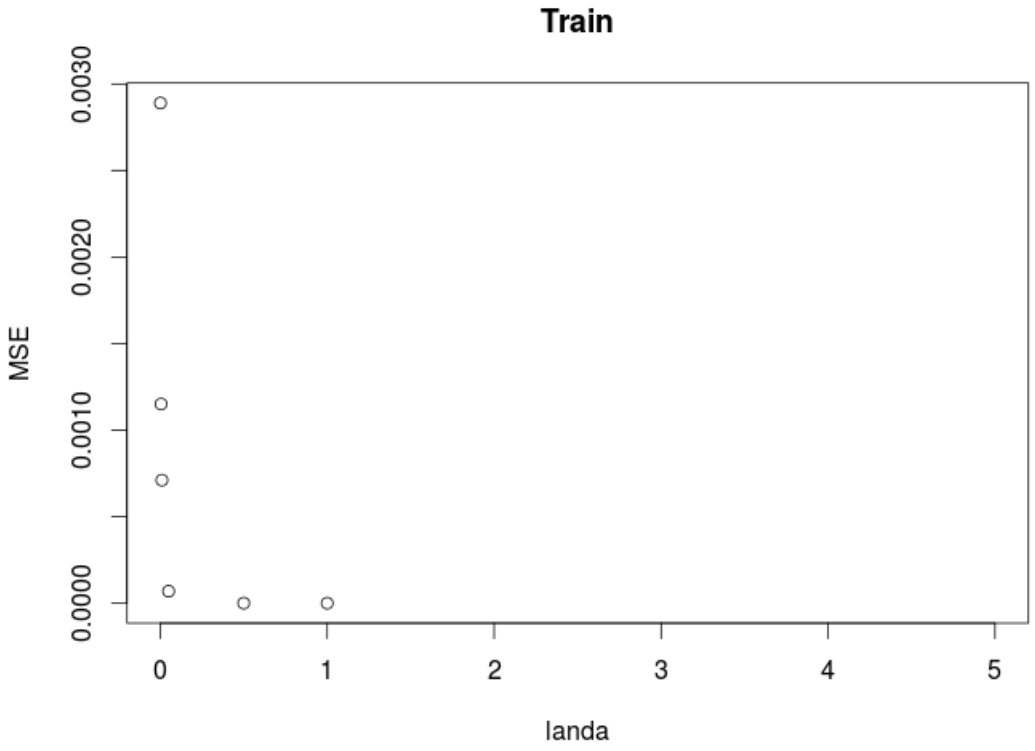

Figura 2.22: Gráfico lambda vs MSE trainig

2.3.2. Realizar el mismo gráfico del punto anterior pero usando los valores de MSE del conjunto de test. Comparar los valores de MSE obtenidos con boosting para el conjunto test con los obtenidos con los métodos de regresión múltiple y LASSO respectivamente para los mismos datos.

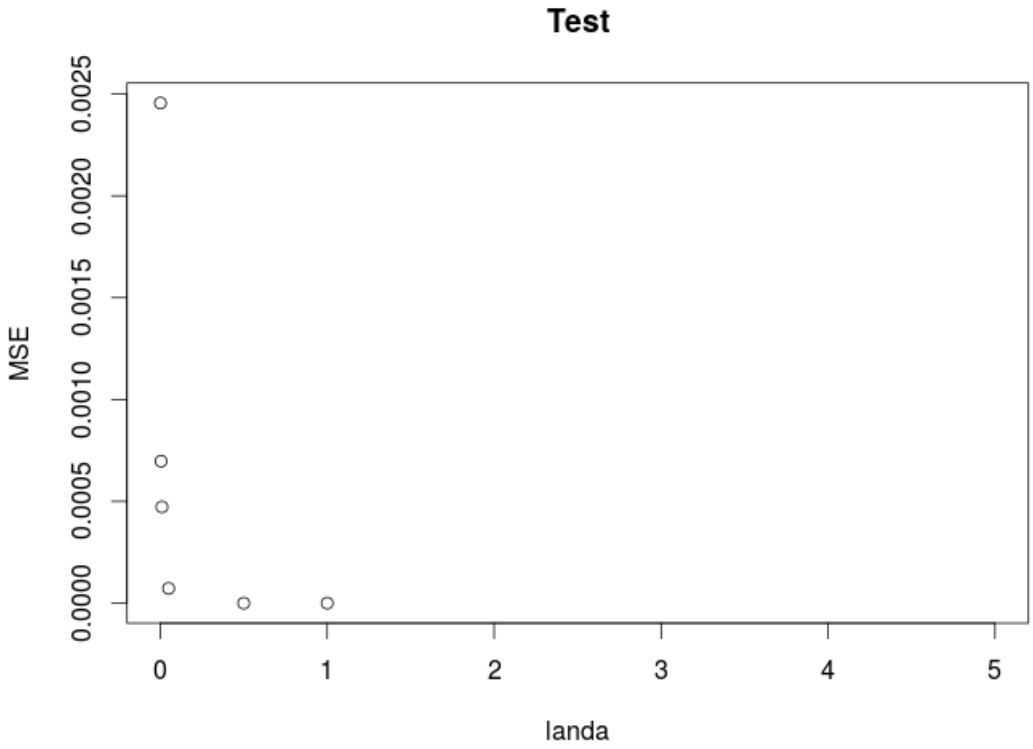

Figura 2.23: Gráfico lambda vs MSE test

2.3.3. ¿Qué variables aparecen como las más importantes en el modelo de "boosting"?

Haciendo summary sobre el modelo obtenemos la siguiente grafica:

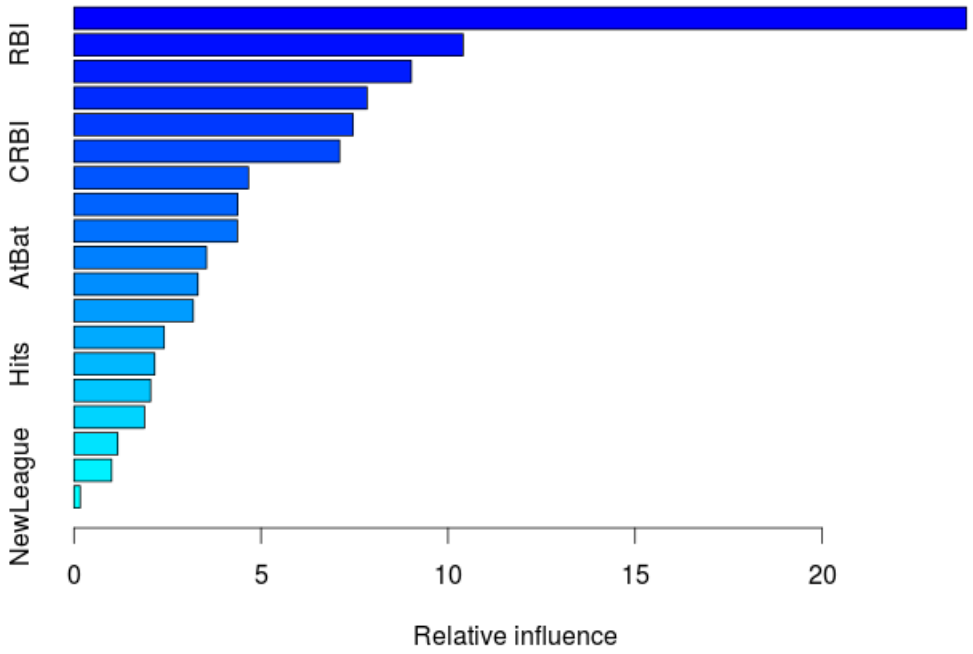

Figura 2.24: summary del modelo boosting

De donde se puede deducir que las variables mas importantes son:

	var	rel.inf
CAtBat	CAtBat	23.8576083
RBI	RBI	10.4043779
PutOuts	PutOuts	9.0175144
Walks	Walks	7.8366364
HmRun	HmRun	7.4631892
CRBI	CRBI	7.1021447

Figura 2.25: summary del modelo boosting

2.3.4. Aplicar bagging al conjunto de "training" y volver a estimar el modelo. ¿Cuál es el valor de MSE para el conjunto de test en este caso?

El valor MSE es de 0.1195, lo que quiere decir que la diferencia entre el estimador y lo que se estima es muy baja debido a que el estimador tiene en cuenta la información que puede producir una estimación más precisa. Aunque este no sea exactamente 0 (se deberá a la aleatoriedad de las muestras), lo es a efectos prácticos.