**Aprendizaje Automático (2014-2015)**

Grado en Ingeniería Informática

Universidad de Granada

# Informe Práctica 2

Mª Cristina Heredia Gómez

2 de mayo de 2015

# Índice

# Índice de figuras

# 1. Ejercicio.-1 (4 puntos): Usar la base de datos Weekly (ISLR) . Este conjunto de datos presenta predicciones semanales (1089) del mercado de valores durante 21 años (1990-2010).

## 1.1. Analizar la conducta de los datos usando resúmenes numéricos y gráficos de los mismos. ¿Se observa algún patrón de interés? En caso afirmativo dar una posible interpretación del mismo.

Podemos comenzar por observar las correlaciones de las variables de este conjunto. Sin embargo, si aplicamos la orden cor() sobre todo el conjunto de datos nos muestra un error diciendo que " debe tener valor numérico´´. Esto se debe a que Direction no es una variable cuantitativa,sino que es cualitativa. Miramos la matriz de correlación quitando esta variable del conjunto:

```
> cor(Weekly[ ,-9])
             Year         Lag1        Lag2        Lag3         Lag4         Lag5      Volume        Today
Year    1.00000000 -0.032289274 -0.03339001 -0.03000649 -0.031127923 -0.030519101  0.84194162 -0.032459894
Lag1   -0.03228927  1.000000000 -0.07485305  0.05863568 -0.071273876 -0.008183096 -0.06495131 -0.075031842
Lag2   -0.03339001 -0.074853051  1.00000000 -0.07572091  0.058381535 -0.072499482 -0.08551314  0.059166717
Lag3   -0.03000649  0.058635682 -0.07572091  1.00000000 -0.075395865  0.060657175 -0.06928771 -0.071243639
Lag4   -0.03112792 -0.071273876  0.05838153 -0.07539587  1.000000000 -0.075675027 -0.06107462 -0.007825873
Lag5   -0.03051910 -0.008183096 -0.07249948  0.06065717 -0.075675027  1.000000000 -0.05851741  0.011012698
Volume  0.84194162 -0.064951313 -0.08551314 -0.06928771 -0.061074617 -0.058517414  1.00000000 -0.033077783
Today  -0.03245989 -0.075031842  0.05916672 -0.07124364 -0.007825873  0.011012698 -0.03307778  1.000000000
```

Figura 1.1: matriz de correlación para todas las variables de Weekly menos Direction

Observamos que la correlación de la variable Today con los Lags está cercana a cero, así como la correlación entre Today y Year o Today y Volume. De aquí podemos deducir que no hay mucha correlación entre el porcentaje devuelto para esta semana (Today) y el devuelto para semanas previas (Lags). Sin embargo, para Year vemos que hay una correlación de 0.84194162 con Volume. Esta correlación sí es significativa. Vamos a pintar el gráfico:

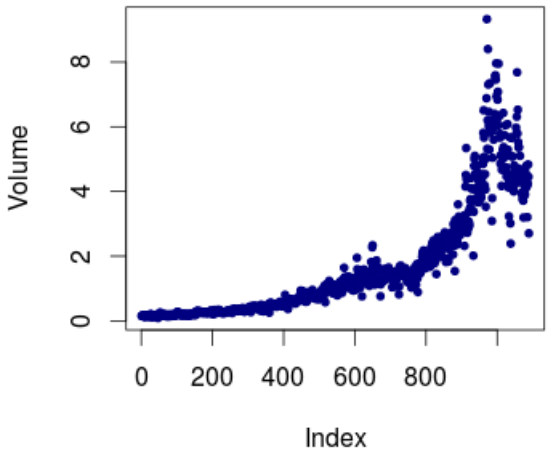


Figura 1.2: gráfico en el que representamos el Volumen

Vemos que según se ve el en gráfico Volume va incrementándose en el tiempo, luego podemos deducir que el número medio de acciones diarias comercializadas fue incrementando desde 1990 a 2010.

## 1.2. Usar la base de datos completa para ajustar un modelo de Regresión Logística usando Direction como variable respuesta y las variables Volume y Lag-1 a Lag-5 como predictores. Usar la función summary() para mostrar los resultados. ¿Alguno de los predictores es estadísticamente significativo? En caso afirmativo, ¿cuál? Justificar la respuesta

```
> glm.fit=glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume, data=Weekly,family=binomial)
> summary(glm.fit)

Call:
glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
    Volume, family = binomial, data = Weekly)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.6949  -1.2565   0.9913   1.0849   1.4579

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.26686    0.08593   3.106   0.0019 **
Lag1        -0.04127    0.02641  -1.563   0.1181
Lag2         0.05844    0.02686   2.175   0.0296 *
Lag3        -0.01606    0.02666  -0.602   0.5469
Lag4        -0.02779    0.02646  -1.050   0.2937
Lag5        -0.01447    0.02638  -0.549   0.5833
Volume      -0.02274    0.03690  -0.616   0.5377
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1496.2  on 1088  degrees of freedom
Residual deviance: 1486.4  on 1082  degrees of freedom
AIC: 1500.4

Number of Fisher Scoring iterations: 4
```

Figura 1.3: Reg.Logística con Direction como response y volume + Lag[1,5] como predictores

Vemos que el único p-valor significativo, es de 0.0296 y corresponde a Lag2. Vemos que el coeficiente que tiene asociado es positivo, por lo que esto podría estar diciéndonos que si el mercado tuvo una “devolución positiva´´ ayer es probable que hoy siga subiendo. Sin embargo, este p-valor tampoco es que sea extremadamente chico por lo que tampoco podemos asegurar que haya relación significativa entre Lag2 y Direction.

## 1.3. Calcular la matriz de confusión y el porcentaje total de predicciones correctas. Explicar lo que la matriz de confusión nos dice acerca de los errores cometidos por regresión logística. Justificar las respuesta

En primer lugar, dados los valores de los predictores, predecimos la probabilidad de que el mercado crezca:

```
> glm.probs=predict(glm.fit, type='response',data=Weekly)
> glm.probs[1:10]
        1         2         3         4         5         6         7         8         9        10
0.6086249 0.6010314 0.5875699 0.4816416 0.6169013 0.5684190 0.5786097 0.5151972 0.5715200 0.5554287
```

Figura 1.4: 10 primeras probabilidades de que el mercado crezca

Ahora convertimos los valores predecidos en vectores de clase con etiquetas "Up´´ o "Down´´. Creamos un vector según si la probabilidad de que el mercado crezca o decrezca es menor o mayor que 0.5:

```
glm.pred=rep("Down",1089)
glm.pred[glm.probs>0.5]="Up"
```

Figura 1.5: vector con etiquetas a üp.º "Down.en función de la probabilidad

y creamos la matriz de confusión y como sabemos que los elementos de la diagonal son las predicciones correctas , sumamos y dividimos por el total de elementos:

```
> table(glm.pred,Direction)
        Direction
glm.pred Down  Up
    Down   54  48
    Up    430 557
> (54+557)/1089
[1] 0.5610652
```

Figura 1.6: matriz de confusión

Obteniendo así que tenemos un **56.11 % de predicciones correctas** en total.
Por otra parte, la matiz de confusión también nos habla de los errores cometidos por la regresión logística, ya que los elementos que quedan fuera de la diagonal, nos indican las predicciones incorrectas. Así pues si sumamos y dividimos por el total, obtenemos (430+48)/1089=0.4389348 ,es decir, un **43.9 % de error**.

## 1.4. Ahora ajustar un modelo de regresión logística a los datos entre 1990 y 2008 usando Lag2 como el único predictor. Calcular la matriz de confusión y la fracción global de predicciones correctas para el periodo 2009 y 2010. Justificar las respuestas

Comenzamos por crear el subconjunto de datos que contenga aquellos datos entre 1990 y 2008. Luego, usando Lag2 como único predictor ajustamos el modelo de regresión logística, especificando con la orden subset que utilizaremos el subconjunto de datos creado anteriormente :

```
> sub_set=(Year<2009)
> glm.fit2=glm(Direction~Lag2, data=Weekly,family=binomial,subset=sub_set)
> summary(glm.fit2)

Call:
glm(formula = Direction ~ Lag2, family = binomial, data = Weekly,
    subset = sub_set)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-1.536  -1.264   1.021   1.091   1.368

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.20326    0.06428   3.162  0.00157 **
Lag2         0.05810    0.02870   2.024  0.04298 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1354.7  on 984  degrees of freedom
Residual deviance: 1350.5  on 983  degrees of freedom
AIC: 1354.5

Number of Fisher Scoring iterations: 4
```

Figura 1.7: Modelo de regresión logística para datos entre 1990 y 2008

Ahora, como se nos pide la matriz de confusión y la fracción global de predicciones correctas para el periodo 2009-2010, en primer lugar tendremos que crear otro vector que contenga las observaciones entre 2009 y 2010. Luego, procedemos de forma similar al apartado anterior, pero teniendo en cuenta que aquí estamos usando dos subconjuntos de datos diferentes para llevar a cabo el entrenamiento y el testeo, pues como conjunto de training tomamos los datos entre 1990-2008, mientras que como conjunto de test usamos los datos tomados entre 2008-2010. Para finalizar, calculamos las predicciones para 2009-1010 y los comparamos con los movimientos que realmente hubo en el mercado durante ese periodo de tiempo:

```
> Weekly.2008=Weekly[!sub_set,]
> length(Weekly.2008)
[1] 9
> dim(Weekly.2008)
[1] 104   9
> Direction.2008=Direction[!sub_set]
> length(Direction.2008)
[1] 104
> glm.probs2=predict(glm.fit2,Weekly.2008, type='response')
> contrasts(Direction)
     Up
Down  0
Up    1
> glm.pred2=rep("Down",104)
> glm.pred2[glm.probs2>0.5]="Up"
> table(glm.pred2,Direction.2008)
         Direction.2008
glm.pred2 Down Up
     Down    9  5
     Up     34 56
```

Figura 1.8: vectores nuevos y matriz de confusión

y haciendo (9+56)/104 obtenemos 0.625. Por lo que el **62 % de los movimientos diarios parecen haber sido bien predecidos**.

```
> (56/(56+34))
[1] 0.6222222
> (9/(9+5))
[1] 0.6428571
```

Figura 1.9: probabilidad de que cuando la RLog. dice que el mercado sube, sube realmente. Y viceversa.

Además, vemos que, haciendo cuentas, la matriz de confusión nos dice que , el día que la regresión logística predice que el mercado sube, esto será verdad un **62 %** de las veces, mientras que cuando la regresión logística predice que el mercado cae, esto será verdad un **64 %** de las veces.

## 1.5. Repetir (4) usando LDA, QDA y KNN con K=1. ¿Cuál de estos métodos parece dar los mejores resultados? Justificar las respuestas.

### 1.5.1. Análisis Discriminantivo Lineal (LDA)

```
> library(ISLR)
> attach(Weekly)
> library(MASS)
> lda.fit =lda( Direction~Lag2 , data=Weekly, subset=sub_set)
> lda.fit
Call:
lda(Direction ~ Lag2, data = Weekly, subset = sub_set)

Prior probabilities of groups:
     Down        Up
0.4477157 0.5522843

Group means:
            Lag2
Down -0.03568254
Up    0.26036581

Coefficients of linear discriminants:
           LD1
Lag2 0.4414162
```

Figura 1.10: Modelo LDA para los datos tomados entre 1990 y 2008

Observamos que en donde pone "Prior probabilities of groups´´ aparece Down=0.4477157 y Up=0.5522843, esto nos dice que el 44.8 % de las observaciones que había en el conjunto de datos de entrenamiento correspondían a días en los que el mercado bajó, mientras que el 55.2 % restante de las observaciones corresponden a el caso contrario. Además, observamos también que en la parte de "group means"tenemos Down=-0.03568254 y Up=0.26036581 , lo que nos sugiere que hay una tendencia a que el porcentaje devuelto en las 2 semanas previas sea positivo cuando el mercado crece , y sea negativo cuando el mercado decrece.

```
> lda.pred = predict( lda.fit, Weekly.2008)
> lda.class = lda.pred$class
> table(lda.class , Direction.2008)
         Direction.2008
lda.class Down Up
     Down    9  5
     Up     34 56
> (9+54)/104
[1] 0.6057692
> mean ( lda.class == Direction.2008)
[1] 0.625
> (9+56)/104
[1] 0.625
```

Figura 1.11: matriz de confusión para LDA

Y observamos que LDA predice esactamente lo mismo que la regresión logística ( un 62 % de predicciones

correctas).

### 1.5.2. Análisis Discriminativo Cuadrático (QDA)

```
> qda.fit = qda (Direction~Lag2 , data =Weekly,  subset = sub_set)
> qda.fit
Call:
qda(Direction ~ Lag2, data = Weekly, subset = sub_set)

Prior probabilities of groups:
     Down        Up
0.4477157 0.5522843

Group means:
            Lag2
Down -0.03568254
Up    0.26036581
```

Figura 1.12: ajuste QDA para datos training (entre 1990 y 2008)

Y vemos que nos muestra los "group means"pero en este caso no se muestran los coeficientes de los discriminantes linales, ya que QDA utiliza una función cuadrática.

```
> qda.class = predict(qda.fit , Weekly.2008)$class
> table(qda.class, Direction.2008)
          Direction.2008
qda.class Down Up
     Down    0  0
     Up     43 61
> mean(qda.class==Direction.2008)
[1] 0.5865385
> 61/104
[1] 0.5865385
```

Figura 1.13: predicción y matriz para QDA

Observamos que en este caso tenemos que un **58.6 % de las predicciones serían correctas**. (Menor procentaje de predicciones correctas que con LDA y con la regresión logística). Yo creo que esto se debe a que el QDA capta de forma más presisa las relaciones entre variables, es decir, que ajusta mejor los datos (en este caso los del conjunto de training, que se corresponden a los datos de entre 1990 y 2008), por lo que al predecir los datos usando el conjunto de datos test da peores resultados que los métodos anteriores.

### 1.5.3. KNN con K=1

```
> set.seed (1)
> train.X = cbind ( Lag2, Lag2 ) [ sub_set ,]
> test.X = cbind ( Lag2, Lag2 ) [! sub_set ,]
> train.Direction = Direction [ sub_set ]
> knn.pred = knn ( train.X , test.X , train.Direction , k =1)
> table ( knn.pred , Direction.2008)
        Direction.2008
knn.pred Down Up
    Down   21 30
    Up     22 31
> mean ( knn.pred == Direction.2008)
[1] 0.5
> (21+31)/104
[1] 0.5
```

Figura 1.14: ajuste KNN con k=1, predicción y matriz

Vemos que en este caso tenemos los peores resultados hasta el momento. Para k=1, tenemos que con este modelo **sólo un 50 % de las predicciones serían correctas**. Esto es porque con K=1 tenemos un ajuste demasiado flexible, por lo que podríamos concluir que, a elegir entre LDA, QDA y KNN con K=1 para estos datos, tendríamos que los mejores modelos son Regresión Logística(apartado anterior) y LDA.

## 1.6. Experimente con diferentes combinaciones de predictores, incluyendo transformaciones de las variables e interacciones entre ellas, en cada uno de los métodos (RLG, LDA, QDA, KNN) (si se desea, pueden usarse técnicas de selección de variables) . Muestre las variables, método y matriz de confusión que da los mejores resultados sobre los datos de test (2009-2010). Justificar las respuestas adecuadamente.

### 1.6.1. RLG

Para RLG lo primero que se nos viene a la mente es usar como predictores Volume y Today, que parecían, en la matriz de correlaciones, ser las más influyentes. Sin embargo, si los tomamos como predictores:

```
> sub_set=(Year<2009)
> glm.fit2=glm(Direction~Volume+Today, data=Weekly,family=binomial,subset=sub_set)
Mensajes de aviso perdidos
1: glm.fit: algorithm did not converge
2: glm.fit: fitted probabilities numerically 0 or 1 occurred
> Weekly.2008=Weekly[!sub_set,]
> dim(Weekly.2008)
[1] 104   9
> Direction.2008=Direction[!sub_set]
> glm.probs2=predict(glm.fit2,Weekly.2008, type='response')
> glm.pred2=rep("Down",104)
> glm.pred2[glm.probs2>0.5]="Up"
> table(glm.pred2,Direction.2008)
         Direction.2008
glm.pred2 Down Up
     Down   43  0
     Up      0 61
> (43+61)/104
[1] 1
```

Figura 1.15: ajuste RLG con Volume y Today como predictores

Vemos que obtenemos un porcentaje de acierto en las predicciones del 100 %. ¿Significa que somos tremendamente buenos ajustando? no. Definitivamente no. Significa que Today no se puede usar como predictor, probablemente porque de ser así estaríamos intentando predecir la respuesta(Direction) con los datos que se emplearon para generarla.
Observamos la matriz de correlaciones a ver si nos da alguna pista:

```
> cor(Weekly[ ,-9])
              Year         Lag1        Lag2        Lag3         Lag4         Lag5      Volume         Today
Year    1.00000000 -0.032289274 -0.03339001 -0.03000649 -0.031127923 -0.030519101  0.84194162 -0.03245989
Lag1   -0.03228927  1.000000000 -0.07485305  0.05863568 -0.071273876 -0.008183096 -0.06495131 -0.07503184
Lag2   -0.03339001 -0.074853051  1.00000000 -0.07572091  0.058381535 -0.072499482 -0.08551314  0.05916671
Lag3   -0.03000649  0.058635682 -0.07572091  1.00000000 -0.075395865  0.060657175 -0.06928771 -0.07124363
Lag4   -0.03112792 -0.071273876  0.05838153 -0.07539587  1.000000000 -0.075675027 -0.06107462 -0.00782587
Lag5   -0.03051910 -0.008183096 -0.07249948  0.06065717 -0.075675027  1.000000000 -0.05851741  0.01101269
Volume  0.84194162 -0.064951313 -0.08551314 -0.06928771 -0.061074617 -0.058517414  1.00000000 -0.03307778
Today  -0.03245989 -0.075031842  0.05916672 -0.07124364 -0.007825873  0.011012698 -0.03307778  1.00000000
```

Figura 1.16: matriz de correlaciones para las variables de Weekly menos Direction

Observamos que existe alguna relación de Lag2 y Lag4, y de Lag3 con Lag1 y Lag5. Pruebo entonces a usar Lag2 con Lag3 como predictores, obteniendo el mismo resultado de porcentaje de aciertos:

```
> glm.fit2=glm(Direction~Lag2+Lag3, data=Weekly,family=binomial,subset=sub_set)
> Weekly.2008=Weekly[!sub_set,]
> Direction.2008=Direction[!sub_set]
> glm.probs2=predict(glm.fit2,Weekly.2008, type='response')
> glm.pred2=rep("Down",104)
> glm.pred2[glm.probs2>0.5]="Up"
> table(glm.pred2,Direction.2008)
         Direction.2008
glm.pred2 Down Up
     Down    8  4
     Up     35 57
> (8+57)/104
[1] 0.625
```

Figura 1.17: ajuste RLG con Lag2 y Lag3 como predictores

Se me ocurre entonces usar Lag2*Lag3 y Lag4 como predictores, obteniendo:

```
> glm.fit2=glm(Direction~Lag2*Lag3+Lag4, data=Weekly,family=binomial,subset=sub_set)
> glm.probs2=predict(glm.fit2,Weekly.2008, type='response')
> glm.pred2=rep("Down",104)
> glm.pred2[glm.probs2>0.5]="Up"
> table(glm.pred2,Direction.2008)
         Direction.2008
glm.pred2 Down Up
     Down    9  4
     Up     34 57
> (9+57)/104
[1] 0.6346154
```

Figura 1.18: ajuste RLG con Lag2,Lag3 y Lag4 como predictores

que vemos que mejora el modelo, aunque no mucho pues ahora tenemos que el **63.4 % de las predicciones serían correctas** frente al 62.5 % que teníamos antes.

### 1.6.2. LDA

Probamos a usar los mismos predictores de antes:

```
> lda.fit =lda( Direction~Lag2*Lag3+Lag4 , data=Weekly, subset=sub_set)
> lda.pred = predict( lda.fit, Weekly.2008)
> lda.class = lda.pred$class
> #creamos matriz de confusión:
> table(lda.class , Direction.2008)
         Direction.2008
lda.class Down Up
     Down    9  4
     Up     34 57
> (9+57)/104
[1] 0.6346154
```

Figura 1.19: ajuste LDA con Lag2,Lag3 y Lag4 como predictores

Y vemos que también mejoramos el porcentaje de predicciones correctas, pasando de 62.5 % a 63.4 % , obteniendo el mismo resultado que con la regresión logística.

### 1.6.3. QDA

Ajustando el modelo usando los mismos predictores de antes (Lag2*Lag3+Lag4) y con la misma respuesta (Direction) obtenemos:

```
> qda.fit=qda(Direction~Lag2*Lag3+Lag4, data =Weekly,subset = sub_set)
> #predecidmos usando modelo y datos test:
> qda.class=predict(qda.fit , Weekly.2008)$class
> #creamos matriz:
> table(qda.class, Direction.2008)
         Direction.2008
qda.class Down Up
     Down    9 11
     Up     34 50
> 59/104
[1] 0.5673077
```

Figura 1.20: ajuste QDA con Lag2,Lag3 y Lag4 como predictores

y vemos que el ajuste empeora usando estos predictores, ya que usando Lag2 como único predictor teníamos que el 58.6 % de las predicciones eran correctas, mientras que ahora sólo el 56.7 % lo son.
Sin embargo, usando como predictores Lag2 y lag3 (Lag2+Lag3), tendríamos que:

```
> qda.fit=qda(Direction~Lag2+Lag3 , data =Weekly,subset = sub_set)
> #predecidmos usando modelo y datos test:
> qda.class=predict(qda.fit , Weekly.2008)$class
> #creamos matriz:
> table(qda.class, Direction.2008)
         Direction.2008
qda.class Down Up
     Down    4  2
     Up     39 59
> (4+59)/104
[1] 0.6057692
```

Figura 1.21: ajuste QDA con Lag2 y Lag3 como predictores

Que sí mejora nuestro modelo QDA inicial en el que usábamos Lag2 como único predictor, pues **obtenemos que un 60.6 %** de las predicciones serían correctas, frente a un 58.6 % que teníamos antes. Sin embargo, si los comparamos con los otros modelos anteriores, tendríamos que con LDA y RGL tendríamos más porcentaje de predicciones correctas.

### 1.6.4. KNN

Probamos por ejemplo, con k=10, y vemos que pasamos de un 50 % de predicciones correctas a un **54.8 %**

```
> library ( class )
> #inicializamos semilla aleatoria
> set.seed(1)
> train.X = cbind ( Lag2, Lag3 ) [ sub_set ,]
> test.X = cbind ( Lag2, Lag3 ) [! sub_set ,]
> train.Direction = Direction [ sub_set ]
> #ajustamos modelo knn:
> knn.pred = knn ( train.X , test.X , train.Direction , k =10)
> #creamos matriz de confusión:
> table ( knn.pred , Direction.2008)
         Direction.2008
knn.pred Down Up
    Down   14 18
    Up     29 43
> (14+43)/104
[1] 0.5480769
```

Figura 1.22: ajuste KNN con Lag2,Lag3 como predictores y k=10

Vemos si con K=12, mejora algo y tenemos:

```
> sub_set=(Year<2009)
> set.seed(1)
> train.X = cbind ( Lag2, Lag3 ) [ sub_set ,]
> test.X = cbind ( Lag2, Lag3 ) [! sub_set ,]
> train.Direction = Direction [ sub_set ]
> #ajustamos modelo knn:
> knn.pred = knn ( train.X , test.X , train.Direction , k =12)
> #creamos matriz de confusión:
> table ( knn.pred , Direction.2008)
        Direction.2008
knn.pred Down Up
    Down   17 13
    Up     26 48
> (17+48)/104
[1] 0.625
```

Figura 1.23: ajuste KNN con Lag2 y Lag3 como predictores y k=12

por lo que pasamos de tener un 54.8 % de predicciones correctas a un **62.5 %** .
Concluimos entonces que los modelos que más predicciones correctas parecen para este conjunto de datos son RLG y LDA, con igual porcentaje de predicciones correctas (63.4 %), usando Direction como response y Lag2*Lag3+Lag4 como predictores.

## 1.7. Repetir el punto anterior usando un modelo de ajuste de validación cruzada con 10 particiones. Comparar con los resultados obtenidos en el punto anterior. Justificar las respuestas adecuadamente.

```
> set.seed(17)
> cv.error.10=rep(0,10)
> for(i in 1:10){
+   glm.fit=glm(Direction~poly(Lag2,i),data=Weekly,family=binomial)
+   cv.error.10[i]=cv.glm(Weekly,glm.fit,K=10)$delta[1]
+ }
Hubo 23 avisos (use warnings() para verlos)
> cv.error.10
 [1] 0.2463549 0.2466795 0.2477240 0.2478395 0.2479929 0.2488988 0.3564807 0.2486654 0.2705604 0.2495251
```

Figura 1.24: ajuste validación cruzada con 10 particiones usando Lag2 como predictor

y observamos en el resultado que obtenemos menor error cuando usamos polinomios de grado uno (modelo lineal) que cuando usamos polinomios de grado dos o más (modelo cuadrático,...) Algo así como nos decía el ejercicio anterior, pues para QDA obteníamos los peores resultados.

## 2. Ejercicio.-2 (3 puntos) En este ejercicio desarrollaremos un modelo para predecir si un coche tiene un consumo de carburante alto o bajo usando la base de datos Auto.

### 2.1. Crear una variable binaria, mpg01, que será igual 1 si mpg contiene un valor por encima de la mediana, y 0 si mpg contiene un valor por debajo de la mediana. La mediana se puede calcular usando la función median(). ( Nota: puede resultar útil usar la función data.frame() para unir en un mismo conjunto de datos mpg01 y las otras variables de Auto)

```
> library(ISLR)
> attach(Auto)
> nrow(Auto)
[1] 392
> mpg01=rep(0,nrow(Auto))
> mpg01[mpg > median(mpg)]=1
> Auto=data.frame(Auto,mpg01)
```

Figura 2.1: crear variable mpg01 y combinarla con las otras variables del conjunto de datos

### 2.2. Explorar los datos gráficamente para investigar la asociación entre mpg01 y las otras características. ¿Cuáles de las otras características parece más útil para predecir mpg01? El uso de Scatterplots y boxplots (ver el libro para recordar concepto) puede resultar útil para contestar la cuestión. Justificar la respuesta.

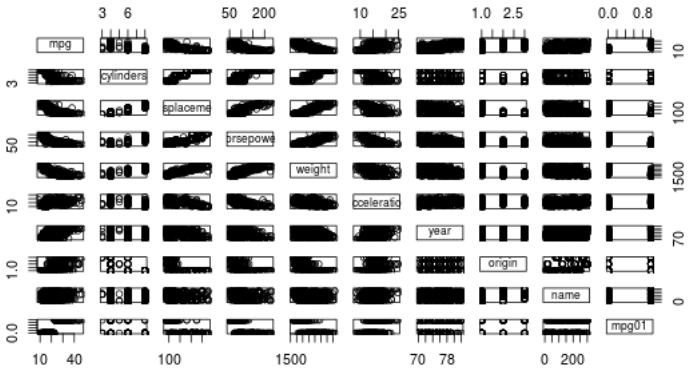


Figura 2.2: parejas de gráficos para las variables del Auto data set + mpg01

Observamos el gráfico por pares. Vemos que entre mpg01 y mpg parece haber relación, pero como hemos utilizado mpg para calcular mpg01 no vamos a centrarnos en ella. Con origin también parece que no hay relación significativa. Pintemos los Boxplots con las demás variables, a ver que nos aclaran:

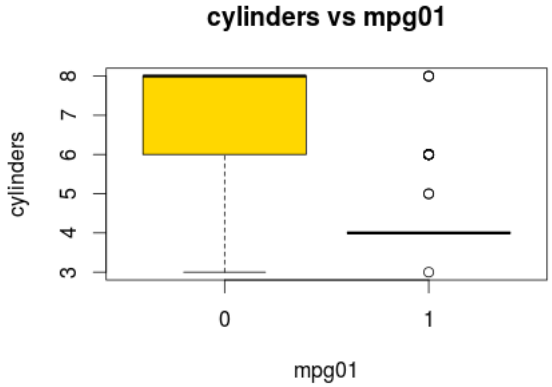


Figura 2.3: boxplot para cylinders vs mpg01

En este caso podemos ver que sí hay asociación entre mpg01 y cylinders, pues todos los valores quedan bien situados por debajo de la mediana, exceptuando unos cuantos que se representan por puntos blancos más grandes y que son los que quedan fuera. (En esencia, los datos en 0 y los datos en 1 no se solapan). Además, puesto que las cajas se muestran bastante separadas entre sí, podemos concluir que la relación entre mpg01 y cylinders es bastante significativa.

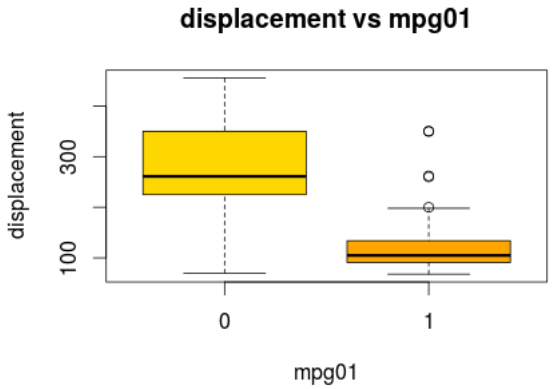


Figura 2.4: boxplot para displacement vs mpg01

En el caso de displacement, también tenemos que existe alguna relación entre mpg01 y displacement, pues para todos los valores, estos quedan bien situados por encima o por debajo de la mediana.(Para los displacement entre 100 y 200, tenemos que quedan por encima de la mediana, mientras que para los

valores de displacement entre 201 y 300 tenemos que quedan por debajo del valor mediano) No hay ninguno que quede por encima y por abajo de la mediana simultáneamente, y que por lo tanto no sepamos a qué parte pertenece. En este gráfico también tenemos algunos valores que varían un poco del resto de muestras. Además, en este caso, también tendríamos que dicha asociación es bastante significativa.

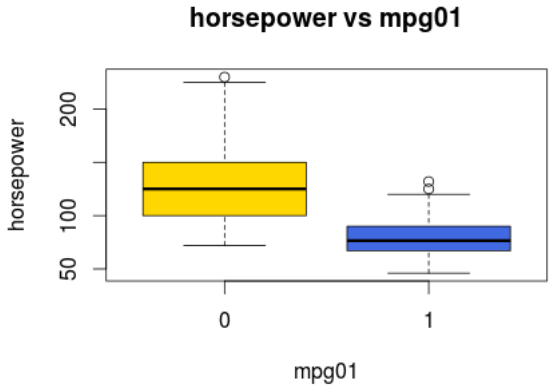


Figura 2.5: boxplot para horsepower vs mpg01

En este caso tenemos que también existe relación entre horsepower y mpg01, pues todos los valores quedan, igualmente, bien situados por encima o por debajo de la mediana.(No se solapan) Para valores de horsepower en el intervalo [100-150] tenemos que quedan por debajo de la media (mpg01=0), mientras que para valores de horsepower en el intervalo [50-100] tenemos que quedan por encima de la mediana(mpg01=1). Sin dejar lugar a dudas de a qué parte pertenece cada uno.

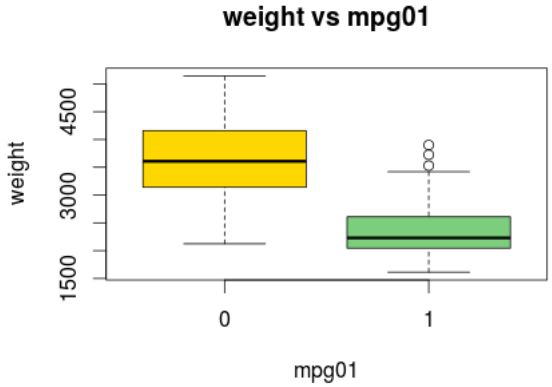


Figura 2.6: boxplot para weight vs mpg01

Con weight tenemos algo parecido al caso anterior. Existe relación entre weight y mpg01, una relación mayor que la horsepower-mpg01, pero menor que la que existía con cylinders-mpg01 y displacement-

mpg01. Para valores de weight en el intervalo[3000-4500] tenemos que quedan por debajo de la media, mientras que para valores de weight en el intervalo[1500-2999] tenemos que quedan por encima de la media.

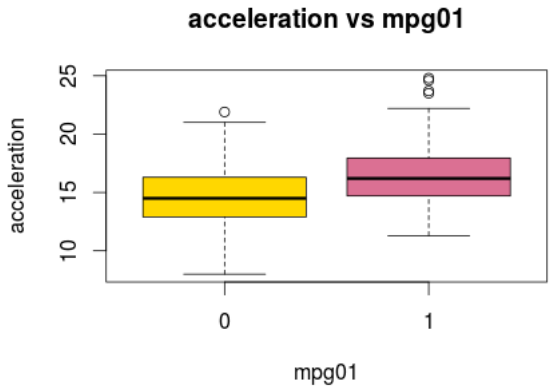


Figura 2.7: boxplot para acceleration vs mpg01

En el gráfico de acceleration vs mpg01 observamos que no existe relación entre estas dos variables, pues vemos que hay valores de acceleration para los que mpg01 vale 0 y 1 simultáneamente,es decir, hay valores en el intervalo [14-17] que quedan por debajo y por encima de la mediana simultáneamente, o dicho de otra manera, las cajitas se solapan.

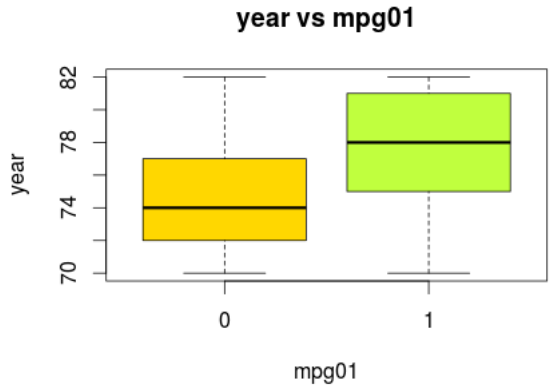


Figura 2.8: boxplot para year vs mpg01

Por una razón análoga a la anterior, tenemos que no existe relación entre year y mpg01, ya que en el gráfico vemos que hay valores de year en el intervalo [75-77] en los que mpg01 vale 0 y 1 simultáneamente, por lo que tenemos valores que en ese intervalo quedan por encima y por debajo de la mediana a la vez. (Se solapan).

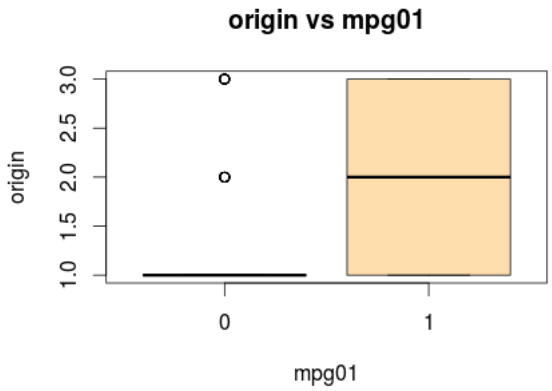


Figura 2.9: boxplot para origin vs mpg01

Vemos que no existe relación relevante entre origin y mpg01, pues para todos los valores de origin, en el intervalo [1.0-3.0] tenemos que todos quedan por encima de la mediana (mpg01=1 siempre).

## 2.3. Definir un conjunto de validación dividiendo los datos en un conjunto de entrenamiento (70 %) y otro de test (30 %):

Primero calculamos cuánto sería el 70 % de los datos, y guardamos este valor en la variable "entrenamiento´´. Una vez que tenemos este porcentaje, le indicamos a la orden sample() que nos divida nuestro conjunto de 392 elementos en 70 % de datos training y 30 % de datos test, de forma aleatoria. Por último, creamos los dos conjuntos y comprobamos que tienen el número de elementos que cada uno debe tener:

```
> entrenamiento=nrow(Auto)*0.7
> train=sample(392,entrenamiento)
> Auto.train=Auto[train, ]
> Auto.test=Auto[-train, ]
> dim(Auto.train)
[1] 274  10
> dim(Auto.test)
[1] 118  10
```

Figura 2.10: definimos conjunto de validación diviendo los datos en dos mitades

### 2.3.1. Ajustar un modelo LDA a los datos de entrenamiento y predecir mpg01 usando las variables que en (b) resultaron más asociadas con mpg01. ¿Cuál es el error de test en el modelo? Justificar la respuesta

```
> lda.fit =lda( mpg01~cylinders+weight+displacement+horsepower, data=Auto, subset=train)
> lda.pred = predict( lda.fit, Auto.test)
> lda.class = lda.pred$class
> length(lda.class)
[1] 118
> mpg01.test=mpg01[-train]
> length(mpg01.test)
[1] 118
> table(lda.class , mpg01.test)
         mpg01.test
lda.class  0  1
        0 48  3
        1 12 55
> (48+55)/118
[1] 0.8728814
> 1-0.8728814
[1] 0.1271186
```

Figura 2.11: Modelo LDA para datos training tomando cylinders,weight,displacement y horsepower como predictores

Observamos que ajustando un modelo LDA para los datos de entrenamiento, al usarlo para predecir los datos test, obtenemos que **un 87.3 % de las predicciones serían correctas**, y que por tanto, el **error de test en el modelo sería de (1- porcentaje acierto), es decir : 12.7 %**.

### 2.3.2. Ajustar un modelo QDA a los datos de entrenamiento y predecir mpg01 usando las variables que en (b) resultaron más asociadas con mpg01. ¿Cuál es el error de test en el modelo? Justificar la respuesta

```
> qda.fit=qda(mpg01~cylinders+weight+displacement+horsepower, data =Auto,subset=train)
> qda.class=predict(qda.fit ,Auto.test)$class
> table(qda.class,mpg01.test)
         mpg01.test
qda.class  0  1
        0 50  4
        1 10 54
> (50+54)/118
[1] 0.8813559
> 1-0.8813559
[1] 0.1186441
```

Figura 2.12: Modelo QDA para datos training tomando cylinders,weight,displacement y horsepower como predictores

Observamos que ajustando un modelo QDA para los datos de entrenamiento, al usarlo para predecir los datos test, obtenemos que un **88.1 % de las predicciones serían correctas**, y que por tanto, **el error de test en el modelo sería de (1 - porcentaje acierto), es decir : 11.9 %** .

### 2.3.3. Ajustar un modelo de regresión Logística a los datos de entrenamiento y predecir mpg01 usando las variables que en (b) resultaron más asociadas con mpg01. ¿Cuál es el error de test en el modelo? Justificar la respuesta

```
> glm.fit2=glm(mpg01~cylinders+weight+displacement+horsepower, data =Auto,family=binomial,subset=train)
> summary(glm.fit2)

Call:
glm(formula = mpg01 ~ cylinders + weight + displacement + horsepower,
    family = binomial, data = Auto, subset = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.2861  -0.2696  -0.0032   0.3806   3.1650

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)  11.2730929  1.9376881   5.818 5.96e-09 ***
cylinders     0.2623756  0.3877166   0.677  0.49858
weight       -0.0020425  0.0008282  -2.466  0.01365 *
displacement -0.0126865  0.0091124  -1.392  0.16385
horsepower   -0.0477432  0.0155837  -3.064  0.00219 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 379.79  on 273  degrees of freedom
Residual deviance: 157.70  on 269  degrees of freedom
AIC: 167.7

Number of Fisher Scoring iterations: 7
```

Figura 2.13: Modelo RLG para datos training tomando cylinders,weight,displacement y horsepower como predictores

```
> glm.probs2=predict(glm.fit2,Auto.test, type='response')
> glm.pred2=rep("Down",118)
> glm.pred2[glm.probs2>0.5]="Up"
> table(glm.pred2,mpg01.test)
         mpg01.test
glm.pred2  0  1
     Down 47  3
     Up   10 58
> (47+58)/118
[1] 0.8898305
> 1-0.8898305
[1] 0.1101695
```

Figura 2.14: Modelo RLG para datos training tomando cylinders,weight,displacement y horsepower como predictores

Observamos que ajustando un modelo RLG para los datos de entrenamiento, al usarlo para predecir los datos test, obtenemos que un **88.9 % de las predicciones serían correctas**, y que por tanto, **el error de test en el modelo sería de (1 - porcentaje acierto), es decir : 11.1 %** .

### 2.3.4. Ajustar un modelo KNN a los datos de entrenamiento y predecir mpg01 usando solamente las variables que en (b) resultaron más asociadas con mpg01. ¿Cuál es el error de test en el modelo? ¿Cuál es el valor de K que mejor ajusta los datos? Justificar la respuesta

```
> set.seed(1)
> train.X = cbind( cylinders,weight,displacement,horsepower )[ train ,]
> test.X = cbind( cylinders,weight,displacement,horsepower )[-train ,]
> train.mpg01 = mpg01[train]
> mpg01.test=mpg01[-train]
> #ajustamos modelo knn:
> knn.pred = knn(train.X , test.X , train.mpg01 , k =1)
> #creamos matriz de confusión:
> table ( knn.pred , mpg01.test)
        mpg01.test
knn.pred  0  1
       0 53 10
       1  6 49
> (53+49)/118
[1] 0.8644068
> 1-0.8644068
[1] 0.1355932
```

Figura 2.15: Modelo KNN para datos training tomando cylinders,weight,displacement y horsepower como predictores y k=1

Observamos que ajustando un modelo KNN para los datos de entrenamiento con k=1, obtenemos que un **86.4 % de las predicciones serían correctas**, y que por tanto, **el error de test en el modelo sería de (1 - porcentaje acierto), es decir : 13.6 %** .

```
> set.seed(1)
> train.X = cbind( cylinders,weight,displacement,horsepower )[ train ,]
> test.X = cbind( cylinders,weight,displacement,horsepower )[-train ,]
> train.mpg01 = mpg01[train]
> mpg01.test=mpg01[-train]
> #ajustamos modelo knn:
> knn.pred = knn(train.X , test.X , train.mpg01 , k =3)
> #creamos matriz de confusión:
> table ( knn.pred , mpg01.test)
        mpg01.test
knn.pred  0  1
       0 53  6
       1  6 53
> (53+53)/118
[1] 0.8983051
> 1-0.8983051
[1] 0.1016949
```

Figura 2.16: Modelo KNN para datos training tomando cylinders,weight,displacement y horsepower como predictores y k=3

Vemos que ajustando un modelo KNN para los datos de entrenamiento con k=3, obtenemos que un **89.8 % de las predicciones serían correctas**, y que por tanto, **el error de test en el modelo sería de (1 - porcentaje acierto), es decir : 10.2 %** , por lo que habríamos obtenido un error menor con k=3. Probemos con k=10:

```
> set.seed(1)
> train.X = cbind( cylinders,weight,displacement,horsepower )[ train ,]
> test.X = cbind( cylinders,weight,displacement,horsepower )[-train ,]
> train.mpg01 = mpg01[train]
> mpg01.test=mpg01[-train]
> #ajustamos modelo knn:
> knn.pred = knn(train.X , test.X , train.mpg01 , k =10)
> #creamos matriz de confusión:
> table ( knn.pred , mpg01.test)
        mpg01.test
knn.pred  0  1
       0 51  6
       1  8 53
> (51+53)/118
[1] 0.8813559
> 1-0.8813559
[1] 0.1186441
```

Figura 2.17: Modelo KNN para datos training tomando cylinders,weight,displacement y horsepower como predictores y k=10

Vemos que ajustando un modelo KNN para los datos de entrenamiento con k=10, obtenemos un **88.13 % de las predicciones serían correctas**, y que por tanto, **el error de test en el modelo sería de (1 - porcentaje acierto), es decir : 11.87 %** por lo que de momento sería mejor el K=3. Por último, vamos a probar con k=100:

```
> set.seed(1)
> train.X = cbind( cylinders,weight,displacement,horsepower )[ train ,]
> test.X = cbind( cylinders,weight,displacement,horsepower )[-train ,]
> train.mpg01 = mpg01[train]
> mpg01.test=mpg01[-train]
> #ajustamos modelo knn:
> knn.pred = knn(train.X , test.X , train.mpg01 , k =100)
> #creamos matriz de confusión:
> table ( knn.pred , mpg01.test)
        mpg01.test
knn.pred  0  1
       0 54  2
       1  5 57
> (54+57)/118
[1] 0.940678
> 1- 0.940678
[1] 0.059322
```

Figura 2.18: Modelo KNN para datos training tomando cylinders,weight,displacement y horsepower como predictores y k=100

Finalmente, observamos que ajustando un modelo KNN para los datos de entrenamiento con k=100, obtenemos que un **94.06 % de las predicciones serían correctas** y que por tanto, **el error de test en el modelo sería de (1 - porcentaje acierto), es decir : 5.94 %** obteniendo los mejores resultados hasta ahora, por lo que 100 parece ser el mejor valor para k ,para predecir los datos.

## 2.4. Repetir los experimentos a-d del punto anterior pero usando Validación Cruzada de 5-particiones para evaluar el error de test. Comparar con los resultados obtenidos en el punto anterior.

```
> set.seed(17)
> cv.error=rep(0,4)
> for(i in 1:4){
+   glm.fit=glm(mpg01~poly(cylinders+weight+displacement+horsepower,i),data=Auto)
+ #  lda.fit =lda( mpg01~poly(cylinders+weight+displacement+horsepower,i), data=Auto,CV=TRUE)
+   cv.error[i]=cv.glm(Auto,glm.fit,K=5)$delta[1]
+ }
> cv.error
[1] 0.10518381 0.09489829 0.09205612 0.08992607
```

Figura 2.19: Validación cruzada con k=5

Usando la misma response y los mismos predictores que en el apartado anterior, podemos ver que con validación cruzada obtenemos que al ajustar un modelo lineal (tipo LDA,por ejemplo) obtendríamos los peores resultados (aproximadamente un 10.5 % de error, mientras que con un modelo cúadrático (como el QDA) podríamos mejorar algo el error, y así con modelos de órdenes superiores. Pero vamos a hacer validación cruzada con k=5 para cada uno de los modelos anteriores. Para ello, hacemos las particiones

.ª mano"dividiendo el conjunto de datos en K(en este caso, 5) subconjuntos. Luego utilizaremos uno de ellos de datos test y el resto serán de training. Esto se repite k iteraciones,para cada posible subconjunto de prueba. Por último, calculamos la media.

### 2.4.1. LDA

```
> errors = rep(0, 5)
> for(i in 0:(K-1)){
+   train = Auto2$id[!(id>=(i*particiones)+1 & id<(particiones*(i+1))+1)]
+   mpg01.test=mpg01[-train]
+   Auto.train=Auto2[train, -ncol(Auto2)]
+   Auto.test=Auto2[-train, -ncol(Auto2)]
+   lda.fit=lda(mpg01~cylinders+weight+displacement+horsepower, data=Auto2[,-ncol(Auto2)], subset=train,
family="binomial")
+   predictor=predict(lda.fit, Auto.test)
+   errors[i+1]=mean(predictor$class != mpg01.test)
+ }
> mean(errors)
[1] 0.1230769
```

Figura 2.20: Validación cruzada con k=5 para lda

Para LDA, tenemos con validación cruzada de k=5 particiones que tendríamos **un error medio de 12.30 %, frente a un 12.7 %** de error que obtuvimos en el apartado anterior para este modelo.

### 2.4.2. QDA

```
> errors = rep(0, 5)
> for(i in 0:(K-1)){
+   train = Auto2$id[!(id>=(i*particiones)+1 & id<(particiones*(i+1))+1)]
+   mpg01.test=mpg01[-train]
+   Auto.train=Auto2[train, -ncol(Auto2)]
+   Auto.test=Auto2[-train, -ncol(Auto2)]
+   qda.fit=qda(mpg01~cylinders+weight+displacement+horsepower, data=Auto2[,-ncol(Auto2)], subset=train,
family="binomial")
+   predictor=predict(qda.fit, Auto.test)
+   errors[i+1]=mean(predictor$class != mpg01.test)
+ }
> mean(errors)
[1] 0.1076923
```

Figura 2.21: Validación cruzada con k=5 para qda

Para QDA, tenemos con validación cruzada de k=5 particiones que tendríamos **un error medio de 10.8 %, frente a un 11.9 %** de error que obtuvimos en el apartado anterior para este modelo.

### 2.4.3. RLG

```
> errors = rep(0, 5)
> for(i in 0:(K-1)){
+   train = Auto2$id[!(id>=(i*particiones)+1 & id<(particiones*(i+1))+1)]
+   mpg01.test=mpg01[-train]
+   Auto.train=Auto2[train, -ncol(Auto2)]
+   Auto.test=Auto2[-train, -ncol(Auto2)]
+   glm.fit=glm(mpg01~cylinders+weight+displacement+horsepower, data=Auto2[,-ncol(Auto2)], subset=train,
family="binomial")
+   probs=predict(glm.fit, Auto.test, type="response")
+   predictor=rep(0, length(probs))
+   predictor[probs > 0.5]=1
+   errors[i+1]=mean(predictor!= mpg01.test)
+ }
> mean(errors)
[1] 0.1205128
```

Figura 2.22: Validación cruzada con k=5 para rlg

Para RLG, tenemos con validación cruzada de k=5 particiones que tendríamos **un error medio de 12.05 %, frente a un 11.1 %** de error que obtuvimos en el apartado anterior para este modelo.

### 2.4.4. KNN

```
> errors = rep(0, 5)
> set.seed(1)
> for(i in 0:4){
+   train = Auto2$id[!(id>=(i*particiones)+1 & id<(particiones*(i+1))+1)]
+   mpg01.test=mpg01[-train]
+   mpg01.train=mpg01[train]
+   Auto.train=Auto2[train, -which(names(Auto2) %in% c("name"))]
+   Auto.test=Auto2[-train, -which(names(Auto2) %in% c("name"))]
+   knn.fit=knn(Auto.train, Auto.test, mpg01.train, k=100)
+   errors[i+1]=mean(knn.fit!= mpg01.test)
+ }
> mean(errors)
[1] 0.1564103
```

Figura 2.23: Validación cruzada con k=5 para KNN

Para KNN con K=100, tenemos que con validación cruzada de k=5 particiones que tendríamos **un error medio de 15.64 %, frente a un 5.94 %** de error que obtuvimos en el apartado anterior para este modelo.

# 3. Ejercicio.-3 (3 puntos) Usar la base de datos Boston para ajustar un modelo que prediga si dado un suburbio este tiene una tasa de criminalidad por encima o por debajo de la mediana. Comparar los modelos encontrados por RLG, LDA y QDA.

## 3.1. Encontrar en cada caso el subconjunto óptimo de variables predictoras usando Validación Cruzada.

### 3.1.1. Escribir una función para el cálculo del error del test.

```
> set.seed(1)
> predict.regsubsets=function(object, newdata, id, ...) {
+   form=as.formula(object$call[[2]])
+   mat=model.matrix(form, newdata)
+   coefi=coef(object, id = id)
+   xvars=names(coefi)
+   mat[, xvars] %*% coefi
+ }
```

Figura 3.1: función para cálculo del error

Ahora que tenemos la función, en primer lugar crearemos un vector donde alojar cada una de las observaciones para las 10 particiones y creamos también una matriz (cv.errors) para guardar los resultados. Inicializamos la semilla:

```
> k = 10
> folds=sample(1:k, nrow(Boston), replace = TRUE)
> cv.errors=matrix(NA, k, 13, dimnames = list(NULL, paste(1:13)))
```

Figura 3.2: creación vector y matriz de resultados

Hacemos la validación cruzada. Tendremos que en la j-ésima partición los elementos con partición==j estarán en el conjunto de test, mientras que los anteriores estarán en el de training. Hacemos las predicciones para cada modelo, calculamos los errores de test, y los almacenamos en su posición correspondiente dentro de la matriz (cv.errors):

```
> for (j in 1:k) {
+   best.fit=regsubsets(crim ~ ., data = Boston[folds != j, ], nvmax = 13)
+   for (i in 1:13) {
+     pred=predict(best.fit, Boston[folds == j, ], id = i)
+     cv.errors[j, i]=mean((Boston$crim[folds == j] - pred)^2)
+   }
+ }
> mean.cv.errors=apply(cv.errors, 2, mean)
> plot(mean.cv.errors, type = "b", xlab = "Number of variables", ylab = "CV error", pch=20, col='orange')
```

Figura 3.3: validación cruzada

Finalmente esto nos da una matriz de 10*13, donde matriz(i,j) es el MSE de test para la partición i-ésima de validación cruzada, para el mejor modelo j-variable. Con apply() podemos obtener el vector que tenga en la posición j el error de la vc para el modelo j-variable. Pintamos los resultados:

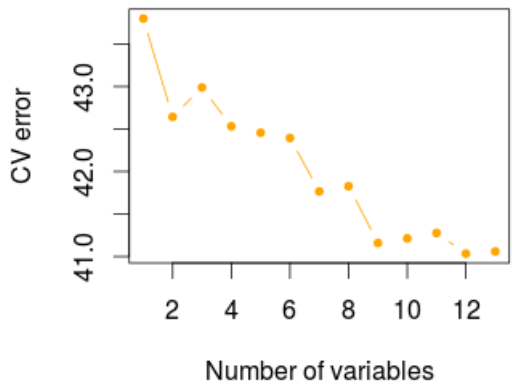


Figura 3.4: modelo de 12 variables seleccionado por validación cruzada

Vemos que la validación cruzada nos ha seleccionado un modelo de 12 variables. Ahora realizamos la selección del mejor subconjunto de variables, dntro de todo el conjuunto de datos, para obtener este modelo:

```
> reg.best = regsubsets( crim ~. , data = Boston , nvmax =13)
> coef ( reg.best ,12)
  (Intercept)            zn         indus          chas           nox            rm           dis
16.985713928   0.044673247  -0.063848469  -0.744367726 -10.202169211   0.439588002  -0.993556631
          rad           tax       ptratio         black         lstat          medv
 0.587660185  -0.003767546  -0.269948860  -0.007518904   0.128120290  -0.198877768
```

Figura 3.5: modelo de 12 variables seleccionado por validación cruzada

Observamos en la gráfica que tenemos una estimación de **error test de 41.2 %** (aprox) para cv.

## 3.2. Calcular los modelos y valorar sus resultados

En primer lugar, como queremos predecir si dado un suburbio este tiene un índice de criminalidad por encima o por debajo de la mediana, procedemos a crear una variable que valga 0 o 1 cuando la criminalidad esté por debajo o por encima de la mediana, respectivamente, y la añadimos a la base de datos de Boston, tal y como hicimos con el conjunto de Auto.

```
> crim01=rep(0,nrow(Boston))
> crim01[crim > median(crim)]=1
> Boston=data.frame(Boston, crim01)
```

Figura 3.6: creación variable crim01 y unión al conjunto Boston

Ahora dividimos el conjunto de datos en la mitad de datos para training y la mitad de datos para test:

```
> test =((nrow(Boston)/2) + 1):nrow(Boston)
> Boston.train=Boston[train, ]
> Boston.test=Boston[test, ]
> crim01.test=crim01[test]
```

Figura 3.7: partimos conjunto en mitad test mitad training

### 3.2.1. RLG

para ajustar el modelo, tomamos como variable response çrim01ż como predictores tomaremos todas las demás, excepto crim01 (pues no vamos a usar la response para predecir la response) y crim (pues la hemos usado para calcular crim01).

```
> glm.probs=predict(glm.fit,Boston.test, type='response')
> glm.pred=rep(0, length(glm.probs))
> glm.pred[glm.probs>0.5]=1
> #creamos matriz de confusión
> table(glm.pred,crim01.test)
        crim01.test
glm.pred   0   1
       0  68  24
       1  22 139
> (68+139)/length(glm.probs)
[1] 0.8181818
```

Figura 3.8: Modelo RLG para datos de Boston data set con crim01 como repsonse

Observamos que ajustando un modelo de regresión logística tomando la mitad de los datos de training y la ota mitad de test, obtenemos que el **81.82 % de las predicciones serían correctas**, mientras que un (1-correctas)= **18.18 % serían erróneas**.

### 3.2.2. LDA

```
> lda.fit =lda(crim01 ~ . - crim01 -crim , data = Boston, family = binomial, subset = train)
> lda.pred = predict( lda.fit, Boston.test)
> lda.class = lda.pred$class
> table(lda.class , crim01.test)
         crim01.test
lda.class   0   1
        0  80  24
        1  10 139
> (80+139)/253
[1] 0.8656126
> 1- 0.8656126
[1] 0.1343874
```

Figura 3.9: Modelo LDA para datos de Boston data set con crim01 como response

Tomando la misma variable response y los mismos predictores que en el caso anterior, observamos que ajustando un modelo LDA, tomando la mitad de los datos de training y la ota mitad de test, obtenemos que el **86.56 % de las predicciones serían correctas**, mientras que un (1-correctas)= **13.44 % serían erróneas**.

### 3.2.3. QDA

```
> qda.fit=qda(crim01 ~ . - crim01 -crim , data = Boston, family = binomial, subset = train)
> qda.class=predict(qda.fit ,Boston.test)$class
> table(qda.class,crim01.test)
         crim01.test
qda.class   0   1
        0  84 159
        1   6   4
> (84+4)/253
[1] 0.3478261
> 1-((84+4)/253)
[1] 0.6521739
```

Figura 3.10: Modelo QDA para datos de Boston data set con crim01 como response

Tomando las mismas variables que hasta el momento, obtenemos los peores resultados de los tres modelos, pues ajustando un model QDA tenemos que tan sólo el **34.78 % de las predicciones serían correctas**, mientras que el **65.22 % serían erróneas**.