Trabajo 2

Aprendizaje Automático

Grado en Ingeniería Informática

Granada, 2 de Mayo de 2015.

Datos del estudiante

Fernández Bosch, Pedro
76422233-H

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.

En este ejercicio se va a examinar la conducta de los datos de Weekly en base a algunos resúmenes numéricos y gráficos.

```
> library(ISLR)
> library (MASS)
> library(class)
> library(boot)
```

```
> data(Weekly)
> attach (Weekly)
```

```
> ?Weekly
```

En primer lugar, es conveniente conocer que esta base de datos contiene 1.089 valores del S&P 500 stock index recogidos semanalmente durante 21 años, desde principios de 1990 hasta finales de 2010.

Para cada fecha/observación, se han registrado los rendimientos porcentuales de los cinco días de cotización anteriores (Lag1 – Lag5). También se ha registrado el volumen de acciones negociadas (Volume), el rendimiento porcentual en la fecha en cuestión (Today) y si el mercado ha subido o bajado en la fecha correspondiente (Direction).

```
> summary (Weekly)
      Year           Lag1               Lag2               Lag3               Lag4               Lag5         
 Min.   :1990   Min.   :-18.1950   Min.   :-18.1950   Min.   :-18.1950   Min.   :-18.1950   Min.   :-18.1950  
 1st Qu.:1995   1st Qu.: -1.1540   1st Qu.: -1.1540   1st Qu.: -1.1580   1st Qu.: -1.1580   1st Qu.: -1.1660  
 Median :2000   Median :  0.2410   Median :  0.2410   Median :  0.2410   Median :  0.2380   Median :  0.2340  
 Mean   :2000   Mean   :  0.1506   Mean   :  0.1511   Mean   :  0.1472   Mean   :  0.1458   Mean   :  0.1399  
 3rd Qu.:2005   3rd Qu.:  1.4050   3rd Qu.:  1.4090   3rd Qu.:  1.4090   3rd Qu.:  1.4090   3rd Qu.:  1.4050  
 Max.   :2010   Max.   : 12.0260   Max.   : 12.0260   Max.   : 12.0260   Max.   : 12.0260   Max.   : 12.0260  
     Volume           Today          Direction  
 Min.   :0.08747   Min.   :-18.1950   Down:484  
 1st Qu.:0.33202   1st Qu.: -1.1540   Up  :605  
 Median :1.00268   Median :  0.2410             
 Mean   :1.57462   Mean   :  0.1499             
 3rd Qu.:2.05373   3rd Qu.:  1.4050             
 Max.   :9.32821   Max.   : 12.0260             
```

Respecto a la variable Year, comprobamos que su rango se encuentra entre 1990 y 2012, ya que su mínimo es 1990 y su máximo es 2010. El valor medio de esta variable es 2000 y el valor de su mediana es 2000.

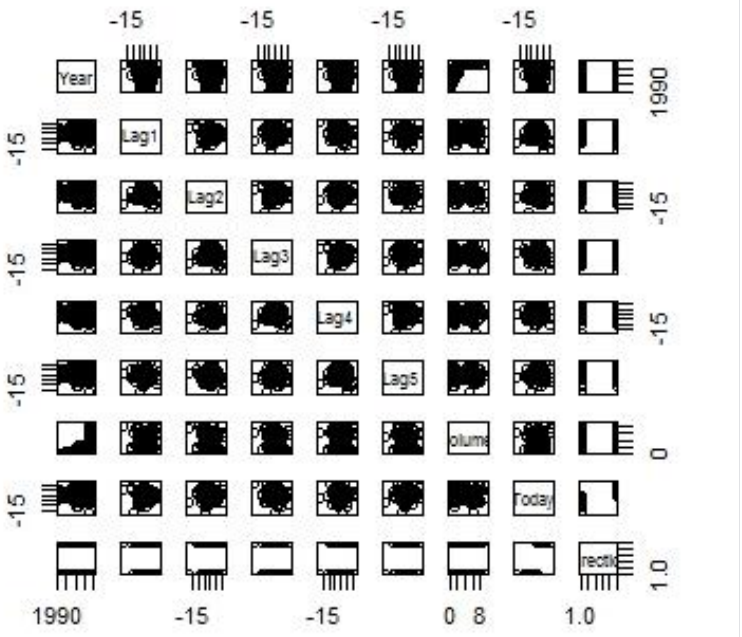
Respecto a las variables Lag1-Lag5 y Today, el valor mínimo para todas ellas es de -18,1950 (rendimiento negativo) y el máximo para todas ellas es de 12,0260 (rendimiento positivo). El valor medio (rendimiento porcentual medio) de estas variables difiere entre Lag1, Lag2, Lag3, Lag4, Lag5 y Today pero mantienen un umbral muy cercano. Así por ejemplo, la mediana para Lag1-Lag3 y Today es de 0,2410, para Lag4 es de 0,2380, para Lag5 es de 0,2340.

Como era de esperar, cada variable tiene un valor medio diferente pero muy cercanos entre si (porque los valores recogidos cada día son diferentes): 0,1506 para Lag1, 0,1511 para Lag2, 0,1472 para Lag3, 0,1458 para Lag4, 0,1399 para Lag5 y 0,1499 para Today.

Respecto a la variable Volume, su mínimo es 0,08747 (volumen mínimo de acciones negociadas) y su máximo es 9,32821 (volumen máximo de acciones negociadas). El valor medio de esta variable es 1,57562 y el valor de su mediana es 1,00268. El valor de la media y la meana distan mucho de su valor máximo, esto quiere decir, que rara vez se alcanza un alto volumen de acciones negociadas.

Y por último, respecto a la variable Direction, es posible afirmar que se registraron 484 bajadas y 605 subidas.

```
> pairs(Weekly)
```

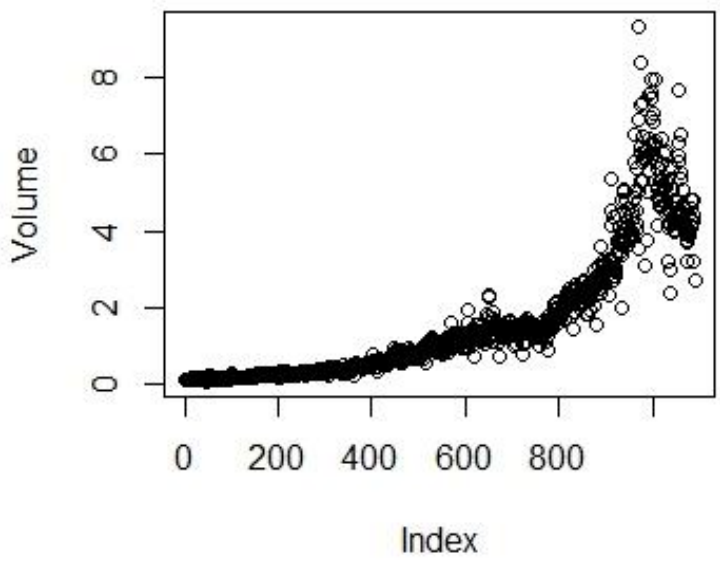


Parece que existen patrones en los datos y relaciones entre Year y Volume, y entre Today y Direction.

```
> 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
```

A partir de la matriz, podemos verificar que existen correlaciones débiles entre la mayoría de las variables, excepto entre Year y Volume.

```
> plot(Volume )
```



Evaluando el gráfico anterior, se puede verificar que Volume aumenta conforme se incrementa el tiempo. En otras palabras, el número promedio de acciones negociadas diariamente aumentó 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

En este ejercicio se va a ajustar un modelo de regresión logística para predecir Direction usando las variables Volume y Lag1 - Lag5 como predictores.

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

Call:
glm(formula = Direction ~ Volume + Lag1 + Lag2 + Lag3 + Lag4 +
    Lag5, 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 **
Volume      -0.02274    0.03690  -0.616   0.5377
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
---
```

```
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
```

Únicamente Lag2 parece ser un predictor estadísticamente significativo, ya que el p-valor obtenido es bastante pequeño.

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 la respuesta.

De acuerdo con el enunciado del problema, para hacer una predicción sobre si el mercado va a subir o bajar en un día particular, es necesario convertir las probabilidades predichas en etiquetas de Up o Down.

```
> glm.pred=rep ("Down", 1089)
> glm.probs =predict (glm.fit ,type ="response")
> glm.pred[glm.probs >.5]="Up"
```

Con los comandos anteriores, se ha creado un vector de 1089 elementos para predecir si la probabilidad de aumento de mercado es mayor (Up) o menor (Down) que 0,5.

A partir de las predicciones obtenidas, se ha creado la matriz de confusión para determinar el número de observaciones que fueron clasificadas correctamente o incorrectamente.

```
> table(glm.pred,Direction)
        Direction
glm.pred Down  Up
    Down   54  48
    Up    430 557
```

Los elementos diagonales de la matriz de confusión indican predicciones correctas, mientras que el resto de elementos representan predicciones incorrectas. De esta matriz se puede deducir que este modelo ha predicho correctamente que el mercado subiría 557 días y que bajaría 54 días, para un total de 557 + 54 = 611 predicciones correctas.

A continuación, se muestra la fracción general de observaciones correctas:

```
> (557+54)/1089
[1] 0.5610652

> mean(glm.pred==Direction)
[1] 0.5610652
```

Los datos obtenidos revelan que el modelo de regresión logística predice correctamente el 56% de los casos.

A simple vista, parece que el modelo de regresión logística tan sólo funciona un poco mejor que una consulta al azar (50%). Los resultados revelan que el porcentaje de predicciones correctas de los datos de entrenamiento es de (54 + 557) / 1089, que es igual al 56,10%. O lo que es lo mismo, la tasa de error de entrenamiento es del 43.89%.

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

Para ajustar el rango de fechas, primero se ha creado un vector correspondiente a los datos desde 1990 hasta 2008.

```
> train=(Year<=2008)
```

Para el conjunto de datos de entrenamiento se ha utilizado el objeto train, un vector de elementos que corresponden a las observaciones de todo el conjunto de datos que se produjeron hasta el año 2008. Las observaciones entre 1990 y 2008 se establecen a true, mientras que las que corresponden a observaciones a partir del año 2009 se establecen a false. Con este método, se ha obtenido una submatriz del conjunto de datos de Weekly relevante para el problema.

```
> train.X=Weekly[train,1:8]
> train.Y=Weekly[train,9]
```

Se ha etiquetado a train.X como una matriz que contiene los predictores asociados a los datos de entrenamiento y a train.Y como un vector que contiene las etiquetas de las observaciones de entrenamiento.

Para el conjunto de datos de test, se ha utilizado el objeto !train, donde los elementos true son intercambiados a false, y los elementos false son intercambiados a true. Por lo tanto, Weekly[!train] produce una submatriz con las observaciones acotadas entre los años 2009 y 2010.

```
> test.X=Weekly[!train,1:8]
> test.Y=Weekly[!train,9]
```

Se ha etiquetado a train.X como una matriz que contiene los predictores asociados a los datos de test sobre los que queremos hacer predicciones y a test.Y como un vector que contiene las etiquetas de las observaciones de test sobre las que queremos hacer predicciones.

```
> glm.fit=glm(Direction~Lag2, data = Weekly[train,], family = binomial)
> summary(glm.fit)

Call:
glm(formula = Direction ~ Lag2, family = binomial, data = Weekly[train,
    ])

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
```

El coeficiente positivo del predictor Lag2, sugiere que si el mercado tuvo un rendimiento positivo durante ese día, entonces es probable que suba en el día actual. Como conclusión, se puede afirmar que un valor 0,04 de p es relativamente pequeño, y por lo tanto existe evidencia de una asociación real entre Lag2 y Direction.

```
> glm.probs=predict(glm.fit, Weekly[!train,], type = "response")
> glm.pred=rep("Down",length(test.Y))
> glm.pred[glm.probs>0.5]="Up"
> table(glm.pred, test.Y)
        test.Y
glm.pred Down Up
    Down    9  5
    Up     34 56
```

A partir de las predicciones obtenidas se ha creado una matriz de confusión para determinar el número de observaciones que fueron clasificadas correctamente o incorrectamente.

Los elementos diagonales de la matriz de confusión indican predicciones correctas, mientras que el resto de elementos representan predicciones incorrectas. De esto podemos deducir que nuestro modelo ha predicho correctamente que el mercado subiría 56 días y que bajaría 9 días, para un total de 56 + 9 = 65 predicciones correctas

```
> mean(glm.pred==test.Y)
[1] 0.625
```

También se ha calculado la fracción de días en los que la predicción era correcta. En este caso, la regresión logística ha predicho correctamente el movimiento del mercado en un 62,5% de los casos.

La fracción de predicciones correctas ha aumentado hasta el 62,5% debido a la supresión de otros predictores que no son Lag2.

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

Método LDA

Ahora se va aplicar LDA sobre las condiciones del problema anterior, utilizando la función lda() que forma parte de la biblioteca MASS.

```
> lda.fit =lda(Direction~Lag2 ,data=Weekly ,subset =train)
> lda.pred=predict(lda.fit, Weekly[!train,])
> lda.class=lda.pred$class
```

```
> table(lda.class, test.Y)
         test.Y
lda.class Down Up
     Down    9  5
     Up     34 56
```

```
> mean(lda.class==test.Y)
[1] 0.625
```

Usando el método LDA, se han obtenido exactamente los mismos resultados que usando el método de regresión logística.

Método QDA

Ahora se va aplicar QDA sobre las condiciones del problema anterior, utilizando la función qda() que forma parte de la biblioteca MASS.

```
> qda.fit =qda(Direction~Lag2 ,data=Weekly ,subset =train)
> qda.pred=predict(qda.fit, Weekly[!train,])
> qda.class=qda.pred$class
```

```
> table(qda.class, test.Y)
         test.Y
qda.class Down Up
     Down    0  0
     Up     43 61
```

Las predicciones obtenidas por el método QDA apuntan a que el mercado siempre va a subir. Por lo tanto, los resultados de esta predicción no son fiables, pues bien sabemos que el mercado de valores tiende a subir y a bajar.

Método KNN con K=1

En último lugar, se va aplicar KNN sobre las condiciones del problema anterior, utilizando la función knn() que forma parte de la biblioteca class.

Antes de nada, con la variable Lag2 se obtiene una matriz para el conjunto de entrenamiento y otra matriz para el conjunto de test.

```
> train.Lag2.X = as.matrix(Lag2[train])
> test.Lag2.X = as.matrix(Lag2[!train])
```

Se ha creado una semilla (seed) aleatoria antes de aplicar la función knn() por si varias observaciones se establecen como vecinos más cercanos, que R pueda decidir al azar el empate. Por lo tanto, la semilla se debe ajustar con el fin de asegurar la reproducibilidad de los resultados.

```
> set.seed(1)
```

A diferencia de los anteriores métodos, KNN es diferente. En lugar de un enfoque de dos etapas en el que primero hay que ajustar el modelo y segundo utilizar el modelo para hacer predicciones, el método knn() forma predicciones usando un solo comando.

Además, el modelo precisa de un valor para K, el número de vecinos más cercanos para ser utilizado por el clasificador. En este caso particular, y según las especificaciones del problema, K tiene valor K=1.

```
> knn.pred=knn(train.Lag2.X,test.Lag2.X,train.Y,k=1)
> table(knn.pred, test.Y)
        test.Y
knn.pred Down Up
    Down   21 30
    Up     22 31
```

```
> mean(knn.pred==test.Y)
[1] 0.5
```

Los resultados utilizando K = 1 no son muy buenos, ya que sólo el 50% de las predicciones serían correctas.

Por lo tanto RGL y LDA son los mejores métodos.

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.

No es posible utilizar Today como predictor:

Today no es un buen predictor, porque Today y Direction contienen información duplicada. Cuando Today tiene valor positivo, Direction tiene valor Up, y viceversa.

```
> Today.Up=Today > 0
> Direction[Today.Up]
  [1] Up Up Up Up Up Up Up Up Up Up Up Up Up Up Up Up Up Up Up Up Up
...

> Today.Down=Today < 0
> Direction[Today.Down]
  [1] Down Down Down Down Down Down Down Down Down Down Down Down Down
...
```

Utilizando Lag4, Lag5 y Volume como predictores:

Método RLG

En primer lugar, se va a aplicar el método RLG sobre los predictores Lag4, Lag5 y Volume.

```
> glm.fit=glm(Direction~Lag4+Lag5+Volume, data = Weekly[train,], family = b
inomial)
> glm.probs=predict(glm.fit, Weekly[!train,], type = "response")
> glm.pred=rep("Down",length(test.Y))
> glm.pred[glm.probs>0.5]="Up"
glm.pred Down Up
    Down   33 44
    Up     10 17
```

Los elementos diagonales de la matriz de confusión indican predicciones correctas, mientras que el resto de elementos representan predicciones incorrectas. De esto podemos deducir que nuestro modelo ha predicho correctamente que el mercado subiría 17 días y que bajaría 33 días, para un total de 17 + 33 = 50 predicciones correctas

```
> mean(glm.pred==test.Y)
[1] 0.4807692
```

En el método de ajuste por Regresión Logística el 48% de las predicciones del mercado de valores serían correctas. Esta cifra es un 10% inferior que el caso del ejercicio previo en el que se utiliza Lag2 como variable predictora.

Método LDA

En esta ocasión, se va a aplicar el método LDA sobre los predictores Lag4, Lag5 y Volume.

```
> lda.fit =lda(Direction~Lag4+Lag5+Volume ,data=Weekly ,subset =train)
> lda.pred=predict(lda.fit, Weekly[!train,])
> lda.class=lda.pred$class
> table(lda.class, test.Y)
         test.Y
lda.class Down Up
     Down   33 44
     Up     10 17

> mean(lda.class==test.Y)
[1] 0.4807692
```

De nuevo, usando el método LDA, se han obtenido exactamente los mismos resultados que usando el método de regresión logística.

Método QDA

En esta ocasión, se va a aplicar el método QDA sobre los predictores Lag4, Lag5 y Volume.

```
> qda.fit =qda(Direction~Lag4+Lag5+Volume,data=Weekly ,subset =train)
> qda.pred=predict(qda.fit, Weekly[!train,])
> qda.class=qda.pred$class
> table(qda.class, test.Y)
         test.Y
qda.class Down Up
     Down   42 59
     Up      1  2

> mean(qda.class==test.Y)
[1] 0.4230769
```

Los datos obtenidos establecen que el modelo QDA sólo predice correctamente el 42% de los casos.

Método KNN

En esta ocasión, se va a aplicar el método KNN sobre los predictores Lag4, Lag5 y Volume.

Antes de nada, se van a combinar las variables Lag4, Lag5 y Volume en una matriz para el conjunto de entrenamiento y otra matriz para el conjunto de test.

```
> train.L4L5V.X=cbind(Lag4 ,Lag5, Volume)[train ,]
> test.L4L5V.X=cbind(Lag4 ,Lag5, Volume)[!train ,]

> set.seed(1)
```

El modelo precisa de un valor para K, el número de vecinos más cercanos para ser utilizado por el clasificador. Primero vamos a probar con un valor de K=1.

```
> knn.pred=knn(train.L4L5V.X,test.L4L5V.X,train.Y,k=1)
> table(knn.pred, test.Y)
        test.Y
knn.pred Down Up
    Down   20 34
    Up     23 27
```

```
> mean(knn.pred==test.Y)
[1] 0.4519231
```

Los datos obtenidos establecen que el modelo KNN con K=1 sólo predice correctamente el 45% de los casos.

Ahora vamos a probar con un valor de K=10.

```
> knn.pred=knn(train.L4L5V.X,test.L4L5V.X,train.Y,k=10)
> table(knn.pred, test.Y)
        test.Y
knn.pred Down Up
    Down   21 27
    Up     22 34

> mean(knn.pred==test.Y)
[1] 0.5288462
```

Los resultados utilizando K = 10 son algo mejores, ya que el número de predicciones correctas ha aumentado hasta el 53%.

Por lo tanto, KNN con K=10 el mejor método en este caso.

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.

En este ejercicio se va a repetir el punto anterior usando un modelo de ajuste de validación cruzada con 10 particiones.

En primer lugar se van a ordenar de forma aleatoria las filas de la base de datos Weekly. Para ello se ha utilizado la función sample() y se ha definido la matriz random.weekly:

```
 nr<-dim(weekly)[1]
 random.weekly<-weekly[sample.int(nr),]
 dim(random.weekly)
[1] 1089    9
```

A continuación, se muestra una visualización esquemática de 10 ciclos de Cross-Validation. Un conjunto de 1.089 observaciones ordenadas aleatoriamente y que se divide en diez grupos que no se solapan. Cada una de estas partes de cada ciclo, actúa como un conjunto de validación (que se muestra en color naranja), y el resto como conjunto de entrenamiento (en azul). El error de la prueba se calcula promediando las diez estimaciones MSE resultantes.

1:108									
	109:216								
		217:324							
			325:432						
				433:540					
					541:648				
						649:756			
							757:864		
								865:972	
									973:1089

Definición de variables:
cv.error es un vector de 10 elementos que almacena el resultado medio del error de test en cada iteración.

a y *b* son las variables de apoyo con las que dividir las observaciones en los subconjuntos de test y train, correspondientemente.

Método RLG

En esta ocasión, se va a repetir el experimento del punto anterior para RGL, pero usando k-Fold Cross-Validation de 10 particiones para evaluar el error de test.

```
> set.seed (17)
> cv.error= rep (0 ,10)
> for (i in 1:10) {
+     glm.fit=glm(Direction~poly(Lag4+Lag5+Volume ,i),data=Weekly, family=b
inomial)
+     cv.error[i]=cv.glm (Weekly ,glm.fit ,K=10) $delta[1]
+ }
> cv.error
 [1] 0.2474423 0.2480334 0.2471058 0.2480511 0.2482423 0.2487527 0.2498678
0.2503265 0.2487343 0.3234421

> mean(cv.error)
[1] 0.2559998
```

El error de test en el modelo RLG usando k-Fold Cross-Validation de 10 particiones es de 25,59%, ha mejorado los resultados del experimento del punto anterior que era de 48,07%.

Método LDA

En esta ocasión, se va a repetir el experimento del punto anterior para LDA, pero usando Validación Cruzada de 5 particiones para evaluar el error de test.

```
> cv.error=rep (0,10)
> a=1
> b=108
>
> for (i in 1:10){
+     test= random.weekly [a:b,]
+     train= random.weekly [-(a:b),]
+
+     lda.fit =lda(Direction~Lag4+Lag5+Volume ,data=train)
+     lda.pred=predict(lda.fit, test)
+     lda.class=lda.pred$class
+
+     cv.error[i]= mean(lda.class!= test[,9])
+
+     a=a+108
+
+     if(i < 10){
+         b=b+108
+     } else{
+         b=b+117
+     }
+ }

> cv.error
 [1] 0.4259259 0.4814815 0.3981481 0.4166667 0.4351852 0.3981481 0.4814815
0.4444444 0.5462963 0.4722222

> mean(cv.error)
[1] 0.45
```

El error de test en el modelo LDA usando Validación Cruzada de 10 particiones es de 45,00%, ha mejorado los resultados del experimento del punto anterior que era de 48,07%.

Método QDA

En esta ocasión, se va a repetir el experimento del punto anterior para QDA, pero usando Validación Cruzada de 10 particiones para evaluar el error de test.

```
> cv.error=rep (0,10)
> a=1
> b=108
>
> for (i in 1:10){
+     test= random.weekly [a:b,]
+     train= random.weekly [-(a:b),]
+
+     qda.fit =qda(Direction~Lag4+Lag5+Volume ,data=train)
+     qda.pred=predict(qda.fit, test)
+     qda.class=qda.pred$class
+
+     mean(qda.class!= test[,9])
+     cv.error[i]= mean(qda.class!= test[,9])
+
+     a=a+108
+
+     if(i < 10){
+         b=b+108
+     } else{
+         b=b+117
+     }
+ }

> cv.error
 [1] 0.4074074 0.4629630 0.4074074 0.3796296 0.3981481 0.4351852 0.4814815
0.4814815 0.5277778 0.4629630

> mean(cv.error)
[1] 0.4444444
```

El error de test en el modelo QDA usando Validación Cruzada de 10 particiones es de 44,44%, ha empeorado respecto a los resultados del experimento del punto anterior que era de 42,30%. Esto puede deberse, a que el resultado del ejercicio anterior es un valor óptimo, es decir, podemos afirmar que Cross Validation ha dado mejores resultados en algunas iteraciones (37,36% en el mejor caso), aunque la media de todas ellas es algo peor.

Método KNN

En esta ocasión, se va a repetir el experimento del punto anterior para KNN con K=10, pero usando Validación Cruzada de 10 particiones para evaluar el error de test.

```
> set.seed(1)
> cv.error=rep (0,10)
> a=1
> b=108
>
> for (i in 1:10){
+     test= random.weekly[a:b,]
+     test.X= test[,c(5,6,7)]
+     test.Y= test[,c(9)]
+     train= random.weekly[-(a:b),]
+     train.X= train[,c(5,6,7)]
+     train.Y= train[,c(9)]
+
+     knn.pred=knn(train.X,test.X, train.Y,k=10)
+     cv.error[i]= mean(knn.pred!= test.Y)
+
```

```
+     a=a+108
+
+     if(i < 10){
+         b=b+108
+     } else{
+         b=b+117
+     }
+ }

> cv.error
 [1] 0.5648148 0.4259259 0.4907407 0.4537037 0.5092593 0.4814815 0.5092593
0.4722222 0.5092593 0.4814815

> mean(cv.error)
[1] 0.4898148
```

El error de test en el modelo KNN con K=10 usando Validación Cruzada de 10 particiones es de 48,98%, ha mejorado los resultados del experimento del punto anterior que era de 52,88%.

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.

```
library (MASS)
library (ISLR)
```

```
> data(Auto)
> attach(Auto)
```

a) 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)

En primer lugar, se ha creado una variable binaria llamada mpg01 inicializada a 0.

```
> Auto$mpg01=0
```

A continuación, todos los valores de mpg que se encuentren por debajo de la mediana, se han actualizado a 1.

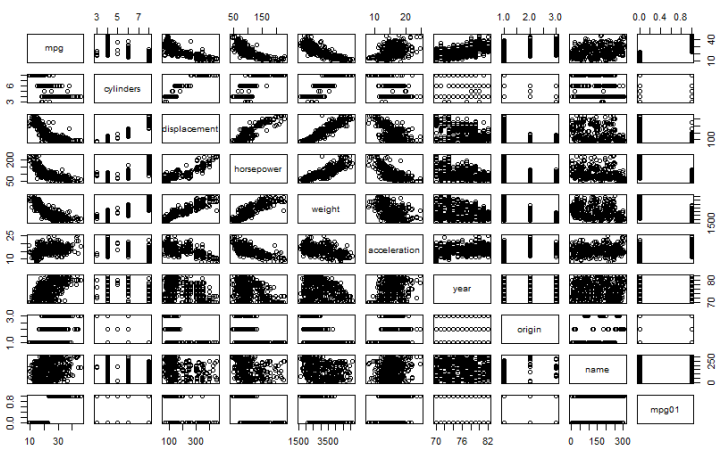
```
> Auto$mpg01[Auto$mpg>median(Auto$mpg)]=1
```

```
> names(Auto)
 [1] "mpg" "cylinders" "displacement" "horsepower" "weight"
 "acceleration" "year" "origin" "name" "mpg01"
```

b) 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.

La función pairs() crea una scatterplot matrix (matriz de dispersión), es decir, un diagrama de dispersión por cada par de variables para cualquier conjunto de datos dado, de esta se puede prever que variables afectarán al consumo. También es posible crear diagramas de dispersión para un solo subconjunto de las variables.

```
> pairs(Auto)
```

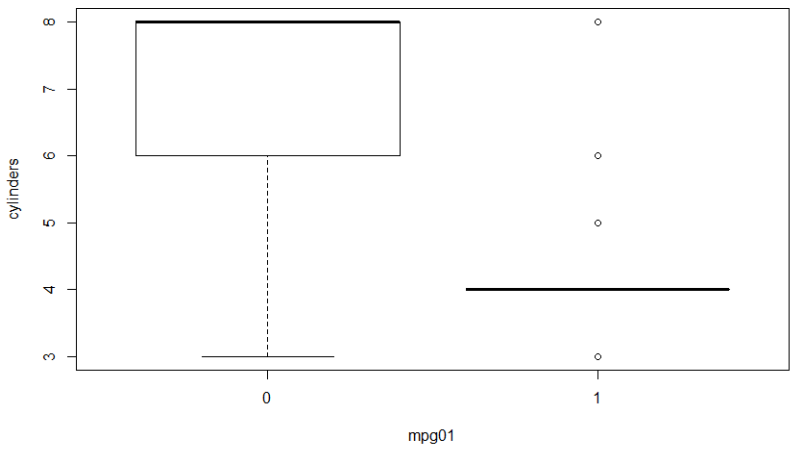


```
> table(Auto$mpg01)
  0   1
196 196
```

Cylinder:

Si la variable representada en el eje x es cualitativa, la función boxplots() puede representarla.

```
> boxplot(cylinders ~ mpg01, data = Auto, xlab="mpg01",ylab ="cylinders")
```



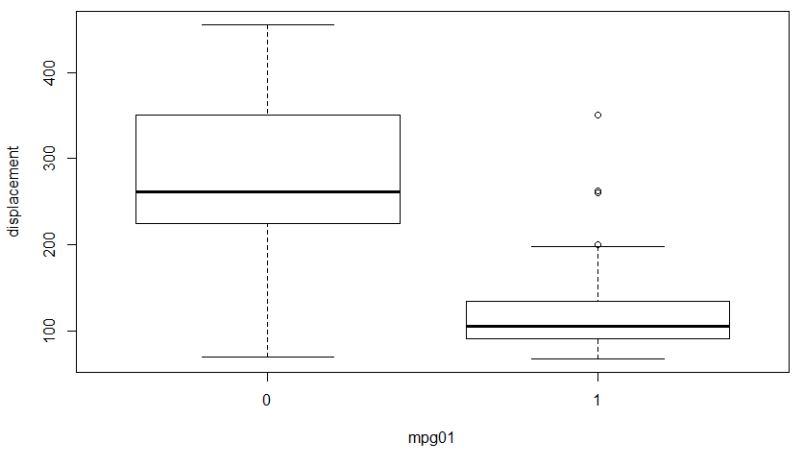
Interpretando la gráfica, es posible deducir que en la base de datos Auto:

Para los coches que tienen 4 cilindros existen más observaciones en mpg01=1. Esto significa que los vehículos con 4, tienden a tener un valor por encima de la mediana de mpg.

Para los coches que tienen 3, 6 y 8 cilindros existen más observaciones en mpg01=0. Esto significa que los vehículos con 3, 6 y 8 cilindros, tienden a tener un valor por debajo de la mediana de mpg.

Displacement:

```
> boxplot(displacement ~ mpg01, data = Auto, xlab="mpg01",ylab ="displaceme
nt")
```



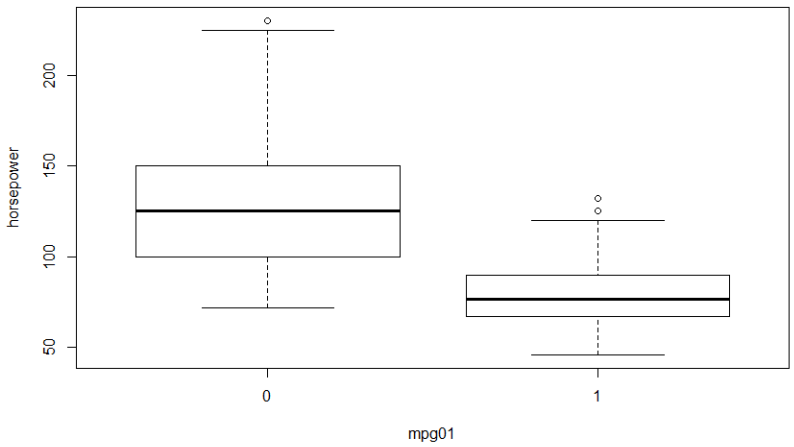
Interpretando la gráfica, es posible deducir que en la base de datos Auto:

Para los vehículos de menor cilindrada del motor (cu. inches), existen más observaciones en mpg01=1. Esto significa que los vehículos con menor cilindrada (cu. inches), tienden a tener un valor por encima de la mediana de mpg.

Para los vehículos de mayor cilindrada del motor (cu. inches), existen más observaciones en mpg01=0. Esto significa que los vehículos con mayor cilindrada (cu. inches), tienden a tener un valor por debajo de la mediana de mpg.

Horsepower:

```
> boxplot(horsepower ~ mpg01, data = Auto, xlab="mpg01",ylab ="horsepower")
```



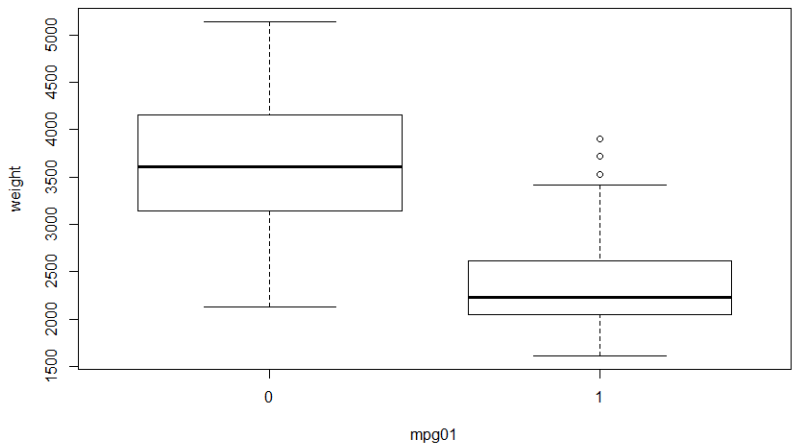
Interpretando la gráfica, se puede deducir que en la base de datos Auto ocurre un efecto similar que el obtenido anteriormente para Displacement:

Para los vehículos con menos caballos de potencia, existen más observaciones en mpg01=1. Esto significa que los vehículos con menos caballos, tienden a tener un valor por encima de la mediana de mpg.

Para los vehículos con más caballos de potencia, existen más observaciones en mpg01=0. Esto significa que los vehículos con más caballos, tienden a tener un valor por debajo de la mediana de mpg.

Weight:

```
> boxplot(weight ~ mpg01, data = Auto, xlab="mpg01",ylab ="weight")
```



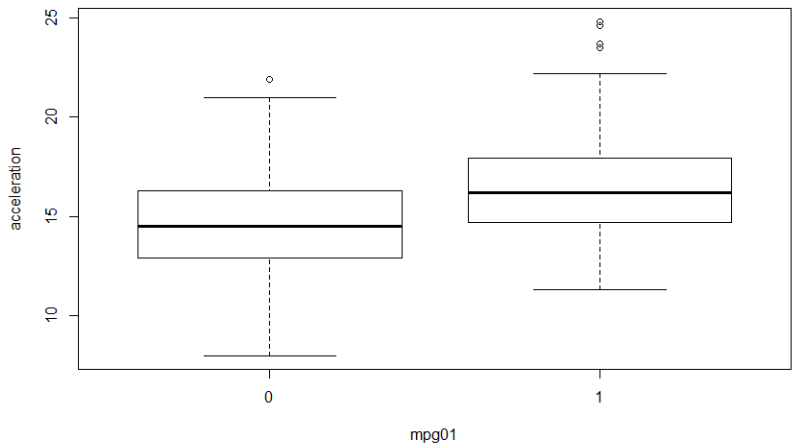
Interpretando la gráfica, se puede deducir que en la base de datos Auto ocurre un efecto similar que el obtenido anteriormente para Displacement y Horsepower:

Para los vehículos con menor peso, existen más observaciones en mpg01=1. Esto significa que los vehículos con menor peso, tienden a tener un valor por encima de la mediana de mpg.

Para los vehículos con mayor peso, existen más observaciones en mpg01=0. Esto significa que los vehículos con mayor peso, tienden a tener un valor por debajo de la mediana de mpg.

Acceleration:

```
> boxplot(acceleration ~ mpg01, data = Auto, xlab="mpg01",ylab ="accelerati
on")
```



La gráfica obtenida no es tan clara como las anteriores, pero podemos intentar deducir que en la base de datos Auto:

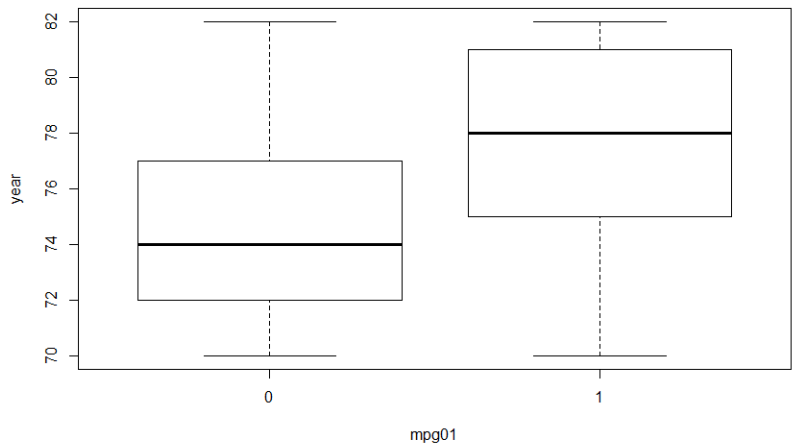
Para los vehículos con aceleración en el intervalo [13-15], se concentran las observaciones en mpg01=0. Esto significa que los vehículos con aceleración baja, tienden a tener un valor por debajo de la mediana de mpg.

Para los vehículos con aceleración en el intervalo [16-18], se concentran las observaciones en mpg01=1. Esto significa que los vehículos con aceleración alta, tienden a tener un valor por encima de la mediana de mpg.

Para el intervalo [15-16] no es posible decidir un tipo de mpg01 predominante, debido a que aproximadamente hay la misma cantidad de observaciones para el valor 0 y 1.

Year:

```
> boxplot(year ~ mpg01, data = Auto, xlab="mpg01",ylab ="year")
```



De nuevo, la gráfica obtenida no es demasiado clara en determinada franja, pero podemos intentar deducir que en la base de datos Auto:

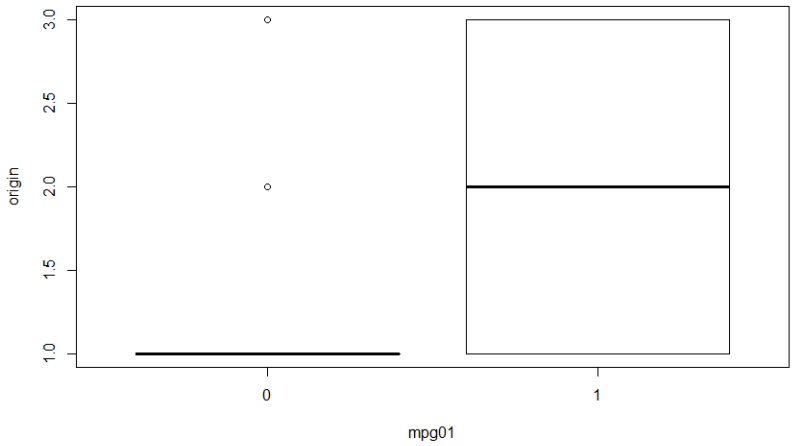
Para los vehículos previos al año 75, existen más puntos en mpg01=0. Esto significa que los vehículos que datan previamente al año 75, tienden a tener un valor por debajo de la mediana de mpg.

Para los vehículos a partir del año 77, existen más puntos en mpg01=1. Esto significa que los vehículos que datan a partir del año 77, tienden a tener un valor por encima de la mediana de mpg.

Para el intervalo de años [75-77] no es posible decidir un tipo de mpg01 predominante, debido a que aproximadamente hay la misma cantidad de observaciones para el valor 0 y 1.

Origin:

```
> boxplot(origin ~ mpg01, data = Auto, xlab="mpg01",ylab ="origin")
```



Interpretando la gráfica, es posible deducir que en la base de datos Auto:

Para los vehículos procedentes el origen 1, existen observaciones en mpg01=0 y mpg01=1. Esto significa que los vehículos procedentes de 1, tienen valores por encima y por debajo de la mediana de mpg.

Para los vehículos procedentes el origen 2 y 3, existen más observaciones en mpg01 = 1. Esto significa que los vehículos procedentes de 2 y 3, tienden a tener observaciones por encima de la mediana de mpg.

Name:

Esta variable no puede aplicarse a este estudio porque es texto aleatorio. Por lo tanto, name se va a excluir del estudio.

A primera vista, parece que cylinders, displacement, horsepower, weight, year y origin son las variables más útiles para predecir mpg01

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

a. 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

La base de datos Auto está compuesta por 392 observaciones, por lo tanto, el 70% para entrenamiento (train) se va a formar por un subconjunto de 275 observaciones y el 30% restante para test se va a formar por un subconjunto de 117 observaciones.

```
> train=1:275
> test=276:392

> names(Auto)
 [1] "mpg"          "cylinders"    "displacement" "horsepower"   "wei
ght"         "acceleration" "year"         "origin"
 [9] "name"         "mpg01"
```

Una vez divididos los datos del conjunto de entrenamiento y test, se van a excluir las variables que en (b) resultaron poco asociadas con mpg01 o que no son relevantes. Estas variables a excluir son: mpg, acceleration, name y mpg10

```
> train.X=Auto[train,c(-1,-6,-9,-10)]
> names(train.X)
[1] "cylinders"    "displacement" "horsepower"   "weight"       "year
"         "origin"

> train.Y=Auto[train,c(10)]
> test.X=Auto[test,c(-1,-6,-9,-10)]
> test.Y=Auto[test,c(10)]

> table(test.Y)
test.Y
 0  1
19 98
```

En esta ocasión, se va a aplicar el método LDA sobre las variables cylinders, displacement, horsepower, weight, year y origin.

```
> lda.fit =lda(mpg01~cylinders+ displacement+horsepower+weight+year+o
rigin ,data= Auto,subset =train)
> lda.pred=predict(lda.fit, test.X)

> lda.class=lda.pred$class

> table(lda.class, test.Y)
         test.Y
lda.class  0  1
        0 18 13
        1  1 85

> mean(lda.class!=test.Y)
[1] 0.1196581
```

El error de test en el modelo LDA es de 11,96%.

b. 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

En esta ocasión, se va a aplicar el método QDA sobre las variables cylinders, displacement, horsepower, weight, year y origin.

```
> qda.fit =qda(mpg01~cylinders+ displacement+horsepower+weight+year+o
rigin ,data= Auto,subset =train)
> qda.pred=predict(qda.fit, test.X)
> qda.class=qda.pred$class
```

```
> table(qda.class, test.Y)
         test.Y
qda.class  0  1
        0 18 19
        1  1 79

> mean(qda.class!=test.Y)
[1] 0.1709402
```

El error de test en el modelo QDA es de 17,09%, mayor que en el modelo LDA.

c. 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.

En esta ocasión, se va a aplicar el método RLG sobre las variables cylinders, displacement, horsepower, weight, year y origin.

```
> glm.fit=glm(mpg01~cylinders+displacement+horsepower+weight+year+ori
gin, data = Auto[train,], family = binomial)
> glm.probs=predict(glm.fit, test.X, type = "response")

> glm.pred=rep(0,length(test.Y))
> glm.pred[glm.probs>0.5]=1
> table(glm.pred, test.Y)
        test.Y
glm.pred  0  1
       0 18 14
       1  1 84

> mean(glm.pred!=test.Y)
[1] 0.1282051
```

El error de test en el modelo RLG es de 12,82%, menor que en el modelo QDA, pero mayor que en el modelo LDA.

d. 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

El modelo precisa de un valor para K, el número de vecinos más cercanos para ser utilizado por el clasificador. Tras varias pruebas, se ha determinado que el mejor valor es K=22.

```
> set.seed(1)
> knn.pred=knn(train.X,test.X,train.Y,k=22)

> table(knn.pred, test.Y)
        test.Y
knn.pred  0  1
       0 19 19
       1  0 79

> mean(knn.pred!=test.Y)
[1] 0.1623932
```

El error de test en el modelo KNN con K=22 es de 16,23%, menor que en el modelo QDA, pero mayor que en los modelos LDA y RGL.

d) 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.

En primer lugar se van a ordenar de forma aleatoria las filas de la base de datos Auto. Para ello se ha utilizado la función sample() y se ha definido la matriz random.auto:

```
set.seed (1)

nr<-dim(Auto)[1]
random.auto<-Auto[sample.int(nr),]
> dim(random.auto)
[1] 392  10
```

A continuación, se muestra una visualización esquemática de 5 ciclos de Cross-Validation. Un conjunto de 392 observaciones ordenadas aleatoriamente y que se divide en cinco grupos que no se solapan. Cada una de estas partes de cada ciclo, actúa como un conjunto de validación (que se muestra en color naranja), y el resto como conjunto de entrenamiento (en azul). El error de la prueba se calcula promediando las cinco estimaciones MSE resultantes.

1:78				
	78:156			
		157:234		
			235:312	
				313:392

Definición de variables:
cv.error es un vector de 5 elementos que almacena el resultado medio del error de test en cada iteración.
a y *b* son las variables de apoyo con las que dividir las observaciones en los subconjuntos de test y train, correspondientemente.

Método LDA

En esta ocasión, se va a repetir el experimento del punto anterior para LDA, pero usando Validación Cruzada de 5 particiones para evaluar el error de test.

```
> cv.error=rep (0,5)
> a=1
> b=78
>
> for (i in 1:5){
+     test= random.auto [a:b,]
+     train= random.auto [-(a:b),]
+
+
+     lda.fit =lda(mpg01~cylinders+displacement+horsepower+weight+year+orig
in ,data=train)
+     lda.pred=predict(lda.fit, test)
+     lda.class=lda.pred$class
+
+     cv.error[i]= mean(lda.class!= test[,10])
+
+     a=a+78
+
```

```
+     if(i < 5){
+         b=b+78
+     } else{
+         b=b+80
+     }
+ }

> cv.error
[1] 0.06410256 0.07692308 0.03846154 0.11538462 0.15384615

> mean(cv.error)
[1] 0.08974359
```

El error de test en el modelo LDA usando Validación Cruzada de 5 particiones es de 8,97%, ha mejorado los resultados del experimento del punto anterior que era de 11,96%.

Método QDA

En esta ocasión, se va a repetir el experimento del punto anterior para QDA, pero usando Validación Cruzada de 5 particiones para evaluar el error de test.

```
> cv.error=rep (0,5)
> a=1
> b=78
>
> for (i in 1:5){
+     test= random.auto [a:b,]
+     train= random.auto [-(a:b),]
+
+
+     qda.fit =qda(mpg01~cylinders+displacement+horsepower+weight+year+orig
in ,data=train)
+     qda.pred=predict(qda.fit, test)
+     qda.class=qda.pred$class
+
+     mean(qda.class!= test[,10])
+     cv.error[i]= mean(qda.class!= test[,10])
+
+     a=a+78
+
+     if(i < 5){
+         b=b+78
+     } else{
+         b=b+80
+     }
+ }

> cv.error
[1] 0.07692308 0.08974359 0.10256410 0.08974359 0.14102564

> mean(cv.error)
[1] 0.1
```

El error de test en el modelo QDA usando Validación Cruzada de 5 particiones es de 10,00%, ha mejorado los resultados del experimento del punto anterior que era de 17,09%.

Método RGL

En esta ocasión, se va a repetir el experimento del punto anterior para RGL, pero usando k-Fold Cross-Validation de 5 particiones para evaluar el error de test.

```
> set.seed (17)
> cv.error= rep (0 ,5)
> for (i in 1:5) {
+     glm.fit=glm(mpg01~poly(cylinders+displacement+horsepower+weight+year+
origin,i),data=Auto)
+     cv.error[i]=cv.glm (Auto ,glm.fit ,K=5) $delta[1]
+ }
> cv.error
[1] 0.10546857 0.09530861 0.09247330 0.09040766 0.09011726
>
> mean(cv.error)
[1] 0.09475508
```

El error de test en el modelo RLG usando k-Fold Cross-Validation de 5 particiones es de 9,47%, ha mejorado los resultados del experimento del punto anterior que era de 12,82%.

Método KNN

En esta ocasión, se va a repetir el experimento del punto anterior para KNN con K=22, pero usando Validación Cruzada de 5 particiones para evaluar el error de test.

```
> set.seed(1)
> cv.error=rep (0,5)
> a=1
> b=78
>
> for (i in 1:5){
+     test= random.auto[a:b,]
+     test.X= test[,c(-1,-6,-9,-10)]
+     test.Y= test[,c(10)]
+     train= random.auto[-(a:b),]
+     train.X= train[,c(-1,-6,-9,-10)]
+     train.Y= train[,c(10)]
+
+     knn.pred=knn(train.X,test.X, train.Y,k=22)
+     cv.error[i]= mean(knn.pred!= test.Y)
+
+     a=a+78
+
+     if(i < 5){
+         b=b+78
+     } else{
+         b=b+80
+     }
+ }
> cv.error
[1] 0.06410256 0.12820513 0.11538462 0.15384615 0.15384615
>
> mean(cv.error)
[1] 0.1230769
```

El error de test en el modelo KNN con K=22 usando Validación Cruzada de 5 particiones es de 12,30%, ha mejorado los resultados del experimento del punto anterior que era de 16,23%.

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.

1. Encontrar en cada caso el subconjunto óptimo de variables predictoras usando Validación Cruzada.
   a. **Escribir una función para el cálculo del error del test.**
2. Calcular los modelos y valorar sus resultados

```
library (ISLR)
```

```
> data(Boston)
> attach(Boston)
```

```
> names(Boston)
 [1] "crim"    "zn"      "indus"   "chas"    "nox"     "rm"      "age"
"dis"     "rad"     "tax"     "ptratio" "black"
[13] "lstat"   "medv"
```

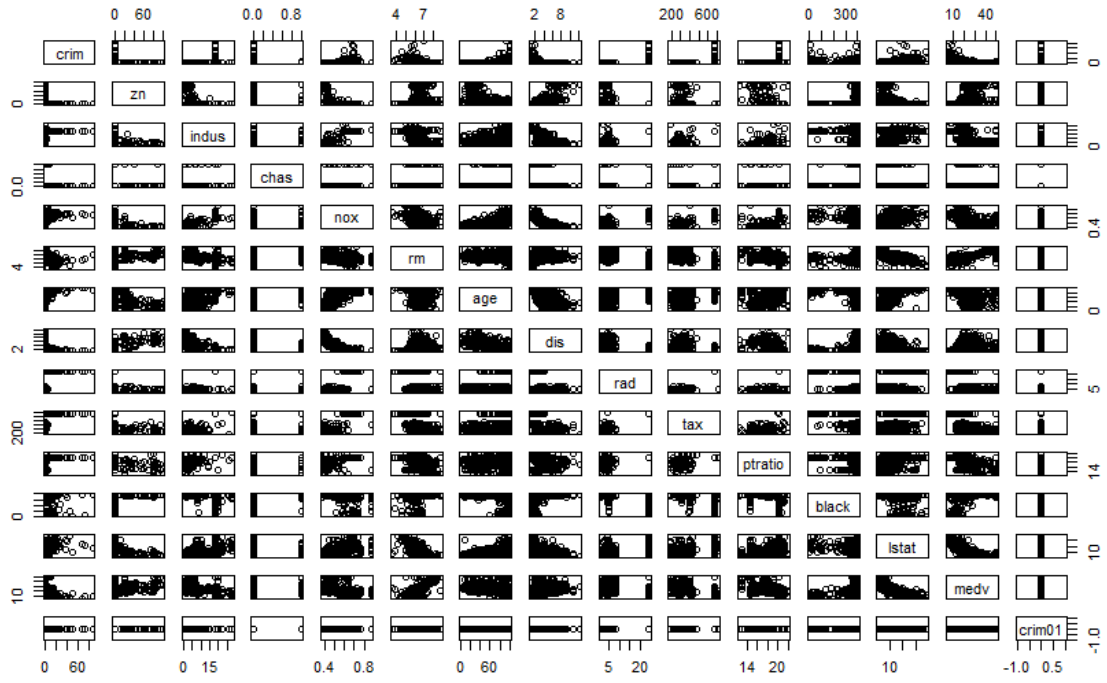
En primer lugar, se ha creado una variable binaria llamada crim01 inicializada a 0.

```
> Boston$crim01=0
```

A continuación, si la tasa de criminalidad de un suburbio está por encima de la mediana, entonces se actualiza su valor a 1

```
> Boston$crim01[crim>median(crim)]=1
```

```
> pairs(Boston)
```




```
> names(Boston)
 [1] "crim"    "zn"      "indus"   "chas"    "nox"     "rm"      "age"
"dis"     "rad"     "tax"     "ptratio" "black"
[13] "lstat"   "medv"    "crim01"
```

Dado que la base de datos de Boston contiene 14 variables, se va a ajustar un modelo de Regresión Logística usando crim01 como variable respuesta y el resto de variables como predictores para obtener el conjunto de los coeficientes de regresión para todos los predictores y elegir los más acertados.

```
> glm.fit=glm(crim01~.,data=Boston)
> summary (glm.fit)

Call:
glm(formula = crim01 ~ ., data = Boston)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-0.6021  -0.2048  -0.0582   0.1588   0.8765

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.4811899  0.3596766  -4.118 4.48e-05 ***
crim         0.0011350  0.0022288   0.509 0.610817
zn          -0.0013810  0.0009315  -1.483 0.138837
indus        0.0030875  0.0041258   0.748 0.454618
chas        -0.0145084  0.0583661  -0.249 0.803793
nox          2.0025143  0.2618145   7.649 1.08e-13 ***
rm           0.0213894  0.0303113   0.706 0.480736
age          0.0027612  0.0008862   3.116 0.001941 **
dis          0.0111019  0.0141047   0.787 0.431598
rad          0.0171097  0.0045460   3.764 0.000188 ***
tax         -0.0002067  0.0002550  -0.810 0.418098
ptratio      0.0125743  0.0092372   1.361 0.174055
black       -0.0002421  0.0001824  -1.328 0.184948
lstat        0.0035227  0.0037541   0.938 0.348518
medv         0.0094707  0.0030244   3.131 0.001843 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for gaussian family taken to be 0.1013345)

    Null deviance: 126.500  on 505  degrees of freedom
Residual deviance:  49.755  on 491  degrees of freedom
AIC: 294.34

Number of Fisher Scoring iterations: 2
```

A la vista de los resultados, ajustando un modelo con el conjunto de todos los predictores, existe una asociación estadísticamente significativa entre predictor y respuesta para los predictores nox, age, rad y medv.

Método RGL

Se va a evaluar el error de test para el modelo RGL con las variables predictoras nox, age, rad y medv usando k-Fold Cross-Validation de 5 particiones.

```
> set.seed (17)
> cv.error= rep (0 ,5)
> for (i in 1:5) {
+     glm.fit=glm(crim01~poly(nox+age+rad+medv,i),data=Boston)
+     cv.error[i]=cv.glm (Boston ,glm.fit ,K=5) $delta[1]
+ }
> cv.error
[1] 0.1394500 0.1347499 0.1302464 0.1282734 0.1276129
>
> mean(cv.error)
[1] 0.1320665
```

El error de test en el modelo RLG usando k-Fold Cross-Validation de 5 particiones es de 13,20%.

A continuación se va a calcular el modelo aplicando el método RLG sobre los predictores nox, age, rad y medv.

```
> glm.fit=glm(crim01~nox+age+rad+medv, data=Boston)
> glm.probs=predict(glm.fit, Boston, type = "response")
> glm.pred=rep(0,length(Boston$crim01))
> glm.pred[glm.probs>0.5]=1
>
> table(glm.pred, Boston$crim01)

glm.pred   0   1
       0 246  60
       1   7 193

> mean(glm.pred== Boston$crim01)
[1] 0.8675889
```

Los datos obtenidos revelan que el modelo de regresión logística predice correctamente el 86,75% de los casos.

Método LDA

En primer lugar se van a ordenar de forma aleatoria las filas de la base de datos Boston. Para ello se ha utilizado la función sample() y se ha definido la matriz random.boston:

```
> nr<-dim(Boston)[1]
> random.boston<-Boston[sample.int(nr),]
> dim(random.boston)
[1] 506  15
```

A continuación, se muestra una visualización esquemática de 5 ciclos de Cross-Validation. Un conjunto de 506 observaciones ordenadas aleatoriamente y que se divide en cinco grupos que no se solapan. Cada una de estas partes de cada ciclo, actúa como un conjunto de validación (que se muestra en color naranja), y el resto como conjunto de entrenamiento (en azul). El error de la prueba se calcula promediando las cinco estimaciones MSE resultantes.

1:101				
	102:202			
		203:303		
			304:405	
				406:506

Definición de variables:
cv.error es un vector de 5 elementos que almacena el resultado medio del error de test en cada iteración.
a y *b* son las variables de apoyo con las que dividir las observaciones en los subconjuntos de test y train, correspondientemente.

En esta ocasión, se va a evaluar el error de test para el modelo LDA con las variables predictoras nox, age, rad y medv usando Cross-Validation de 5 particiones.

```
> cv.error=rep (0,5)
> a=1
> b=101
>
> for (i in 1:5){
+     test= random.boston [a:b,]
```

```
+     train= random.boston [-(a:b),]
+
+
+     lda.fit =lda(crim01~nox+age+rad+medv ,data=train)
+     lda.pred=predict(lda.fit, test)
+     lda.class=lda.pred$class
+
+     cv.error[i]= mean(lda.class!= test[,15])
+
+     a=a+101
+
+     if(i < 5){
+         b=b+101
+     } else{
+         b=b+102
+     }
+ }
> cv.error
[1] 0.1089109 0.1386139 0.1584158 0.1782178 0.1287129
>
> mean(cv.error)
[1] 0.1425743
```

El error de test en el modelo LDA usando Cross-Validation de 5 particiones es de 14,25%, algo mayor que el obtenido con RLG.

A continuación se va a calcular el modelo aplicando el método LDA sobre los predictores nox, age, rad y medv.

```
> lda.fit =lda(crim01~nox+age+rad+medv ,data= Boston)
> lda.pred=predict(lda.fit, Boston)
> lda.class=lda.pred$class
>
> table(lda.class, Boston$crim01)
         crim01
lda.class   0   1
        0 246  60
        1   7 193
>
> mean(lda.class== Boston$crim01)
[1] 0.8675889
```

Los datos obtenidos revelan que el modelo de regresión logística predice correctamente el 86,75% de los casos, igual que con el método de RLG.

Método QDA

En esta ocasión, se va a repetir el experimento del punto anterior para QDA, pero usando Validación Cruzada de 5 particiones para evaluar el error de test.

```
> cv.error=rep (0,5)
> a=1
> b=101
>
> for (i in 1:5){
+     test= random.boston [a:b,]
+     train= random.boston [-(a:b),]
+
+
+     qda.fit =qda(crim01~nox+age+rad+medv ,data=train)
+     qda.pred=predict(qda.fit, test)
+     qda.class=qda.pred$class
+
```

```
+     cv.error[i]= mean(qda.class!= test[,15])
+
+     a=a+101
+
+     if(i < 5){
+         b=b+101
+     } else{
+         b=b+102
+     }
+ }
> cv.error
[1] 0.11881188 0.16831683 0.07920792 0.18811881 0.15841584
>
> mean(cv.error)
[1] 0.1425743
```

El error de test en el modelo QDA usando Cross-Validation de 5 particiones es de 14,25%, igual que el obtenido con LDA y algo mayor que el obtenido con RLG.

A continuación se va a calcular el modelo aplicando el método QDA sobre los predictores nox, age, rad y medv.

```
> qda.fit =qda(crim01~nox+age+rad+medv ,data=Boston)
> qda.pred=predict(qda.fit, Boston)
> qda.class=qda.pred$class
>
> table(qda.class, Boston$crim01)
         crim01
qda.class   0   1
        0 245  56
        1   8 197
>
> mean(qda.class == Boston$crim01)
[1] 0.8735178
```

Los datos obtenidos revelan que el modelo de regresión logística predice correctamente el 87,35% de los casos, mejorando las predicciones de RLG y LDA.

Por lo tanto, en consecuencia de los hechos se deduce que el mejor modelo que predice si dado un suburbio este tiene una tasa de criminalidad por encima o por debajo de la mediana es QDA, utilizando las variables predictoras nox, age, rad y medv.