Tarea 1 - 1EST17

Tabla de Contenidos

Grupo

- **Integrante**: Lucio Enrique Cornejo Ramírez
- **Código**: 20192058

Problema 1

```
library(ISLR2)
```

```
Warning: package 'ISLR2' was built under R version 4.1.3
```

```r
data(Weekly)

head(Weekly)
```

```
  Year   Lag1   Lag2   Lag3   Lag4   Lag5    Volume  Today Direction
1 1990  0.816  1.572 -3.936 -0.229 -3.484 0.1549760 -0.270      Down
2 1990 -0.270  0.816  1.572 -3.936 -0.229 0.1485740 -2.576      Down
3 1990 -2.576 -0.270  0.816  1.572 -3.936 0.1598375  3.514        Up
4 1990  3.514 -2.576 -0.270  0.816  1.572 0.1616300  0.712        Up
5 1990  0.712  3.514 -2.576 -0.270  0.816 0.1537280  1.178        Up
6 1990  1.178  0.712  3.514 -2.576 -0.270 0.1544440 -1.372      Down
```

```r
str(Weekly)
```

```
'data.frame':   1089 obs. of  9 variables:
 $ Year     : num  1990 1990 1990 1990 1990 1990 1990 1990 1990 1990 ...
 $ Lag1     : num  0.816 -0.27 -2.576 3.514 0.712 ...
 $ Lag2     : num  1.572 0.816 -0.27 -2.576 3.514 ...
 $ Lag3     : num  -3.936 1.572 0.816 -0.27 -2.576 ...
 $ Lag4     : num  -0.229 -3.936 1.572 0.816 -0.27 ...
 $ Lag5     : num  -3.484 -0.229 -3.936 1.572 0.816 ...
 $ Volume   : num  0.155 0.149 0.16 0.162 0.154 ...
 $ Today    : num  -0.27 -2.576 3.514 0.712 1.178 ...
 $ Direction: Factor w/ 2 levels "Down","Up": 1 1 2 2 2 1 2 2 2 1 ...
```

Trataremos la categoría `Up` como valor *exitoso* de la variable `Direction` por predecir.

```r
Weekly$Direction <- relevel(Weekly$Direction, "Down")
head(Weekly)
```

```
  Year   Lag1   Lag2   Lag3   Lag4   Lag5    Volume  Today Direction
1 1990  0.816  1.572 -3.936 -0.229 -3.484 0.1549760 -0.270      Down
2 1990 -0.270  0.816  1.572 -3.936 -0.229 0.1485740 -2.576      Down
3 1990 -2.576 -0.270  0.816  1.572 -3.936 0.1598375  3.514        Up
4 1990  3.514 -2.576 -0.270  0.816  1.572 0.1616300  0.712        Up
5 1990  0.712  3.514 -2.576 -0.270  0.816 0.1537280  1.178        Up
6 1990  1.178  0.712  3.514 -2.576 -0.270 0.1544440 -1.372      Down
```

Pregunta a

```r
modelo_logistic <- glm(
  Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume,
  family = binomial, data = Weekly
)
summary(modelo_logistic)
```

```
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
```

De la columna `Pr(>|z|)` en la tabla previa, notamos que solo la variable `Lag2` parece ser estadísticamente significativa en el modelo, puesto que presenta un p-valor asociado de `0.0296`, menor que **0.05**.

Pregunta b

```r
# Probabilidades predichas
yprob <- predict(modelo_logistic, type = 'response')
head(yprob)
```

```
        1         2         3         4         5         6
0.6086249 0.6010314 0.5875699 0.4816416 0.6169013 0.5684190
```

```r
# Valores predichos
ypred <- factor(as.numeric(yprob >= 0.5), labels = levels(Weekly$Direction))
head(ypred)
```

```
[1] Up   Up   Up   Down Up   Up
Levels: Down Up
```

```r
# Matriz de confusión
tmp <- caret::confusionMatrix(
  data = ypred, reference = Weekly$Direction,
  # Fijamos la categoría "Up" de la variable
  # por predecir (Direction), como el valor de éxito
  positive = "Up", mode = "everything"
)
```

```
          Reference
Prediction  Up Down
      Up   557  430
      Down  48   54

$overall
      Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
  5.610652e-01   3.499327e-02   5.310002e-01   5.907994e-01   5.555556e-01
AccuracyPValue  McnemarPValue
  3.690448e-01   5.189745e-68

$byClass
         Sensitivity          Specificity       Pos Pred Value
           0.9206612           0.1115702            0.5643364
```

```
      Neg Pred Value            Precision               Recall
           0.5294118            0.5643364            0.9206612
                   F1           Prevalence       Detection Rate
           0.6997487            0.5555556            0.5114784
Detection Prevalence    Balanced Accuracy
           0.9063361            0.5161157
```

Interpretación de las métricas halladas:

Métrica	Valor	Interpretación
Accuracy	0.5611	Aproximadamente 56% de las observaciones fueron clasificadas correctamente por el modelo.
Coeficiente Kappa	0.035	Como el coeficiente de Kappa es positivo, pero tan cercano a cero, concluimos que el **grado de acuerdo** entre los valores de reales y los predichos por el modelo es insignificante. Es decir, las predicciones del modelo son casi tan precisas como si se predijese por simple azar.
Sensibilidad	0.9207	El modelo predice correctamente aproximadamente 92% de los casos positivos (mercado tuvo rendimiento positivo en una semana) .
Especificidad	0.1116	El modelo predice correctamente aproximadamente 11% de los casos negativos (mercado tuvo rendimiento negativo en una semana) .

Métrica	Valor	Interpretación
Índice de Youden	0.0323	Como el índide de Youden es tan cercano a cero, esto implica que el modelo genera aproximadamente la misma proporción de predicciones *positivas/éxito* para observaciones exitosas y observaciones no exitosas. Es decir, este modelo es prácticamente inútil.
Precisión	0.5643	Este modelo produce una probabilidad de aproximadamente 56% de que una observación predicha como positiva/exitosa, realmente sea positiva/exitosa.
Prevalencia	0.5556	Se estima que, en la población asociada al conjunto de datos `Weekly`, aproximadamente 56% de las instancias sean positivas (rendimiento positivo semanal del mercado).
PPV	0.5643	Según este modelo, la probabilidad de que una observación sea positiva (mercado haya tenido rendimiento positivo en una semana arbitraria entre 1990 y 2010) es de aproximadamente 56% .

Métrica	Valor	Interpretación
NPV	0.5294	Según este modelo, la probabilidad de que una observación sea negativa (mercado haya tenido rendimiento negativo en una semana arbitraria entre 1990 y 2010) es de aproximadamente 53% .

Pregunta c

```
library(pROC)
```

```
Warning: package 'pROC' was built under R version 4.1.3
```

```
Type 'citation("pROC")' for a citation.
```

```
Attaching package: 'pROC'
```

```
The following objects are masked from 'package:stats':

    cov, smooth, var
```

```
levels(Weekly$Direction)
```

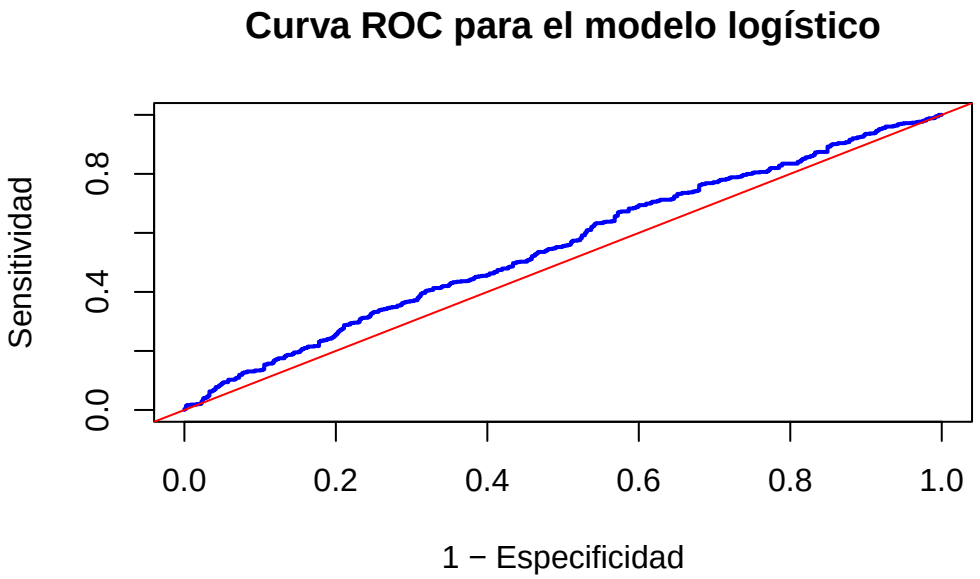
```
[1] "Down" "Up"
```

```
analysis <- roc(response = Weekly$Direction, predictor = yprob)
```

```
Setting levels: control = Down, case = Up
```

```
Setting direction: controls < cases
```

```r
# Curva ROC
plot(
  1 - analysis$specificities, analysis$sensitivities,
  ylab = "Sensitividad", xlab = "1 - Especificidad",
  main = "Curva ROC para el modelo logístico",
  type = "l", col = "blue", lwd = 2
)
abline(a = 0, b = 1, col = "red")
```



Notamos que la curva ROC del modelo está **muy alejada** del caso ideal. Es decir, los puntos de la curva ROC del modelo están lejos de la esquina superior izquierda, la cual representa el caso de un modelo perfecto, donde la sensitividad y especificidad son máximas (valen 1).

```r
# Área bajo la curva
analysis$auc
```

```
Area under the curve: 0.5537
```

Como el área bajo la curva ROC es de 0.5537, un valor muy cercano a 0.5, concluimos que este modelo **no predice adecuadamente** si una observación arbitraria posee un valor positivo/exitoso de clase/Direction.

Pregunta d

```r
# Hallamos el punto de corte óptimo para un balance
# entre sensibilidad y especificidad del modelo
e <- cbind(
  analysis$thresholds,
  analysis$sensitivities + analysis$specificities - 1
)
head(e)
```

```
          [,1]          [,2]
[1,]      -Inf  0.0000000000
[2,] 0.2986107  0.0020661157
[3,] 0.3281353  0.0041322314
[4,] 0.3525944  0.0024793388
[5,] 0.3778156  0.0008264463
[6,] 0.3992318 -0.0008264463
```

```r
opt_t <- subset(e, e[,2] == max(e[,2]))[,1]
opt_t
```

```
[1] 0.5436289
```

```r
ypred2 <- factor(as.numeric(yprob >= opt_t ), labels = levels(Weekly$Direction))
```

```r
# Nueva matriz de confusión
tmp2 <- caret::confusionMatrix(
  data = ypred2, reference = Weekly$Direction,
  positive = "Up", mode = "everything"
)
```

```
          Reference
Prediction  Up Down
      Up   405  277
      Down 200  207
```

Comparamos las métricas del nuevo modelo (solo cambiamos el punto de corte):

```r
metricas_modelos <- rbind(
  data.frame(as.list(c(tmp[[3]], tmp[[4]])), row.names = "Modelo 1"),
  data.frame(as.list(c(tmp2[[3]], tmp2[[4]])), row.names = "Modelo 2")
)
knitr::kable(t(metricas_modelos))
```

	Modelo 1	Modelo 2
Accuracy	0.5610652	0.5619835
Kappa	0.0349933	0.0986773
AccuracyLower	0.5310002	0.5319228
AccuracyUpper	0.5907994	0.5917084
AccuracyNull	0.5555556	0.5555556
AccuracyPValue	0.3690448	0.3462681
McnemarPValue	0.0000000	0.0005018
Sensitivity	0.9206612	0.6694215
Specificity	0.1115702	0.4276860
Pos.Pred.Value	0.5643364	0.5938416
Neg.Pred.Value	0.5294118	0.5085995
Precision	0.5643364	0.5938416
Recall	0.9206612	0.6694215
F1	0.6997487	0.6293706
Prevalence	0.5555556	0.5555556
Detection.Rate	0.5114784	0.3719008
Detection.Prevalence	0.9063361	0.6262626
Balanced.Accuracy	0.5161157	0.5485537

Conclusiones respecto a la tabla previa:

- El segundo modelo posee una **ligeramente mayor accuracy**.
- Pese al incremento del coeficiente de Kappa, el nuevo valor, 0.099 es aún tan cercano a cero que el **grado de acuerdo** entre los valores reales y predichos por el nuevo modelo es aún **insignificante**.
- Debido al balance entre sensibilidad y especificidad, el nuevo modelo posee una **mayor especificidad**, pero **menor sensibilidad**.
- Ambos modelos poseen la **misma prevalencia**, puesto que este valor depende de los datos (muestra); no del modelo.
- Como se está realizando una clasificación **binaria**, el **aumento de PPV** produjo un **menor NPV** para el nuevo modelo.

Pregunta e

```r
summary(Weekly$Year)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1990    1995    2000    2000    2005    2010
```

```r
split_train_validation <- Weekly$Year < 2009
```

```r
# Esquema de validación
training_set <- Weekly[split_train_validation,]
head(training_set)
```

```
  Year   Lag1   Lag2   Lag3   Lag4   Lag5    Volume  Today Direction
1 1990  0.816  1.572 -3.936 -0.229 -3.484 0.1549760 -0.270      Down
2 1990 -0.270  0.816  1.572 -3.936 -0.229 0.1485740 -2.576      Down
3 1990 -2.576 -0.270  0.816  1.572 -3.936 0.1598375  3.514        Up
4 1990  3.514 -2.576 -0.270  0.816  1.572 0.1616300  0.712        Up
5 1990  0.712  3.514 -2.576 -0.270  0.816 0.1537280  1.178        Up
6 1990  1.178  0.712  3.514 -2.576 -0.270 0.1544440 -1.372      Down
```

```r
validation_set <- Weekly[!split_train_validation,]
head(validation_set)
```

```
    Year   Lag1   Lag2   Lag3   Lag4   Lag5   Volume  Today Direction
986 2009  6.760 -1.698  0.926  0.418 -2.251 3.793110 -4.448      Down
987 2009 -4.448  6.760 -1.698  0.926  0.418 5.043904 -4.518      Down
988 2009 -4.518 -4.448  6.760 -1.698  0.926 5.948758 -2.137      Down
989 2009 -2.137 -4.518 -4.448  6.760 -1.698 6.129763 -0.730      Down
990 2009 -0.730 -2.137 -4.518 -4.448  6.760 5.602004  5.173        Up
991 2009  5.173 -0.730 -2.137 -4.518 -4.448 6.217632 -4.808      Down
```

```r
modelo_logistic3 <- glm(
  Direction ~ Lag2, family = binomial, data = training_set
)
summary(modelo_logistic3)
```

```
Call:
glm(formula = Direction ~ Lag2, family = binomial, data = training_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
```

```r
# Probabilidades predichas
yprob3 <- predict(
  modelo_logistic3, newdata = validation_set, type = 'response'
)
head(yprob3)
```

```
      986       987       988       989       990       991
0.5261291 0.6447364 0.4862159 0.4852001 0.5197667 0.5401255
```

```r
# Valores predichos
ypred3 <- as.numeric(yprob3 >= 0.5) |>
  factor(labels = levels(validation_set$Direction))
head(ypred3)
```

```
[1] Up   Up   Down Down Up   Up
Levels: Down Up
```

```r
# Matriz de confusión
tmp3 <- caret::confusionMatrix(
  data = ypred3, reference = validation_set$Direction,
  positive = "Up", mode = "everything"
)
```

```
          Reference
Prediction Up Down
      Up   56   34
      Down  5    9
```

```r
metricas_modelos <- rbind(
  metricas_modelos,
  data.frame(as.list(c(tmp3[[3]], tmp3[[4]])), row.names = "Modelo 3")
)
knitr::kable(t(metricas_modelos))
```

	Modelo 1	Modelo 2	Modelo 3
Accuracy	0.5610652	0.5619835	0.6250000
Kappa	0.0349933	0.0986773	0.1414056
AccuracyLower	0.5310002	0.5319228	0.5246597
AccuracyUpper	0.5907994	0.5917084	0.7180252
AccuracyNull	0.5555556	0.5555556	0.5865385
AccuracyPValue	0.3690448	0.3462681	0.2439500
McnemarPValue	0.0000000	0.0005018	0.0000073
Sensitivity	0.9206612	0.6694215	0.9180328
Specificity	0.1115702	0.4276860	0.2093023
Pos.Pred.Value	0.5643364	0.5938416	0.6222222
Neg.Pred.Value	0.5294118	0.5085995	0.6428571
Precision	0.5643364	0.5938416	0.6222222
Recall	0.9206612	0.6694215	0.9180328
F1	0.6997487	0.6293706	0.7417219
Prevalence	0.5555556	0.5555556	0.5865385
Detection.Rate	0.5114784	0.3719008	0.5384615
Detection.Prevalence	0.9063361	0.6262626	0.8653846
Balanced.Accuracy	0.5161157	0.5485537	0.5636676

Conclusiones:

- El nuevo modelo tiene **mayor accuracy** que los modelos previos, con un valor de 62.5% .
- El nuevo modelo presenta **mayor coeficiente de Kappa** que los modelos previos, aunque aún tan cercano a cero que no hay concordancia entre clases observadas y las pronosticadas.
- El nuevo modelo es mejor que el segundo modelo en predecir una clase que positiva/exitosa ("Up") para observaciones positivas/exitosas; aunque no tan bueno como el primer modelo, pues presenta **menor sensibilidad**.
- En base a los valores de especificidad, el nuevo modelo es mejor que el primer modelo, aunque no tanto como el segundo modelo, en predecir una clase negativa ("Down") para observaciones no exitosas ("Down" como valor de `Direction`).
- En comparación a los dos primeros modelos, el nuevo modelo estima mayores probabilidades de que una nueva observación sea positiva/exitosa (PPV), y de que una nueva observación sea negativa (NPV).

Pregunta f

```r
#:::::::::::::::::::::::::::::::::::: #
# Regresión binaria con enlace probit #
#:::::::::::::::::::::::::::::::::::: #
modelo_4_probit <- glm(
  Direction ~ Lag2, family = binomial(link = "probit"), data = training_set
)
summary(modelo_4_probit)
```

```
Call:
glm(formula = Direction ~ Lag2, family = binomial(link = "probit"),
    data = training_set)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-1.540  -1.264   1.020   1.091   1.369

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.12728    0.04016   3.169  0.00153 **
Lag2         0.03640    0.01783   2.042  0.04120 *
---
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: 3
```

```r
# Probabilidades predichas
yprob4 <- predict(
  modelo_4_probit, newdata = validation_set, type = 'response'
)
# Valores predichos
ypred4 <- as.numeric(yprob4 >= 0.5) |>
  factor(labels = levels(validation_set$Direction))

# Matriz de confusión
tmp4 <- caret::confusionMatrix(
  data = ypred4, reference = validation_set$Direction,
  positive = "Up", mode = "everything"
)

metricas_modelos <- rbind(
  metricas_modelos,
  data.frame(as.list(c(tmp4[[3]], tmp4[[4]])), row.names = "Modelo 4: probit")
)
```

```
          Reference
Prediction Up Down
      Up   56   34
      Down  5    9
```

```r
#:::::::::::::::::::::::::::::::::::: #
# Regresión binaria con enlace cloglog #
#:::::::::::::::::::::::::::::::::::: #
modelo_5_cloglog <- glm(
  Direction ~ Lag2, family = binomial(link = "cloglog"), data = training_set
)
summary(modelo_5_cloglog)
```

```
Call:
```

```
glm(formula = Direction ~ Lag2, family = binomial(link = "cloglog"),
    data = training_set)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-1.575  -1.262   1.017   1.091   1.363

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.22485    0.04433  -5.072 3.94e-07 ***
Lag2         0.04197    0.01983   2.116   0.0343 *
---
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: 1350.4  on 983  degrees of freedom
AIC: 1354.4

Number of Fisher Scoring iterations: 5
```

```r
# Probabilidades predichas
yprob5 <- predict(
  modelo_5_cloglog, newdata = validation_set, type = 'response'
)
# Valores predichos
ypred5 <- as.numeric(yprob5 >= 0.5) |>
  factor(labels = levels(validation_set$Direction))

# Matriz de confusión
tmp5 <- caret::confusionMatrix(
  data = ypred5, reference = validation_set$Direction,
  positive = "Up", mode = "everything"
)

metricas_modelos <- rbind(
  metricas_modelos,
  data.frame(as.list(c(tmp5[[3]], tmp5[[4]])), row.names = "Modelo 5: cloglog")
)
```

```
          Reference
```

```
Prediction Up Down
      Up   56   34
      Down  5    9
```

```r
#:::::::::::::::::::::::::::::::::::: #
# Regresión binaria con enlace cauchit #
#:::::::::::::::::::::::::::::::::::: #
modelo_6_cauchit <- glm(
  Direction ~ Lag2, family = binomial(link = "cauchit"), data = training_set
)
summary(modelo_6_cauchit)
```

```
Call:
glm(formula = Direction ~ Lag2, family = binomial(link = "cauchit"),
    data = training_set)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-1.512  -1.264   1.023   1.091   1.364

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.16041    0.05151   3.114  0.00185 **
Lag2         0.04573    0.02370   1.929  0.05369 .
---
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: 1350.7  on 983  degrees of freedom
AIC: 1354.7

Number of Fisher Scoring iterations: 4
```

```r
# Probabilidades predichas
yprob6 <- predict(
  modelo_6_cauchit, newdata = validation_set, type = 'response'
)
# Valores predichos
ypred6 <- as.numeric(yprob6 >= 0.5) |>
```

```
  factor(labels = levels(validation_set$Direction))

# Matriz de confusión
tmp6 <- caret::confusionMatrix(
  data = ypred6, reference = validation_set$Direction,
  positive = "Up", mode = "everything"
)

metricas_modelos <- rbind(
  metricas_modelos,
  data.frame(as.list(c(tmp6[[3]], tmp6[[4]])), row.names = "Modelo 6: cauchit")
)
```

```
          Reference
Prediction Up Down
      Up   56   34
      Down  5    9
```

```
#::::::::::::::::::::::::::::::::::::::::::::: #
# Modelo Naive Bayes con estimación no
# paramétrica de la densidad del predictor Lag2
#::::::::::::::::::::::::::::::::::::::::::::: #
library(naivebayes)
```

```
Warning: package 'naivebayes' was built under R version 4.1.3
```

```
naivebayes 0.9.7 loaded
```

```
modelo_7_naive_bayes <- naive_bayes(
  x = training_set["Lag2"], y = training_set$Direction,
  usekernel = TRUE
)

# Probabilidades predichas
yprob7 <- predict(modelo_7_naive_bayes, newdata = validation_set["Lag2"], type = 'prob')[,

# Valores predichos
ypred7 <- predict(modelo_7_naive_bayes, newdata = validation_set["Lag2"])
```

```r
# Matriz de confusión
tmp7 <- caret::confusionMatrix(
  data = ypred7, reference = validation_set$Direction,
  positive = "Up", mode = "everything"
)

metricas_modelos <- rbind(
  metricas_modelos,
  data.frame(as.list(c(tmp7[[3]], tmp7[[4]])), row.names = "Modelo 7: Naive Bayes")
)
```

```
          Reference
Prediction Up Down
      Up   56   36
      Down  5    7
```

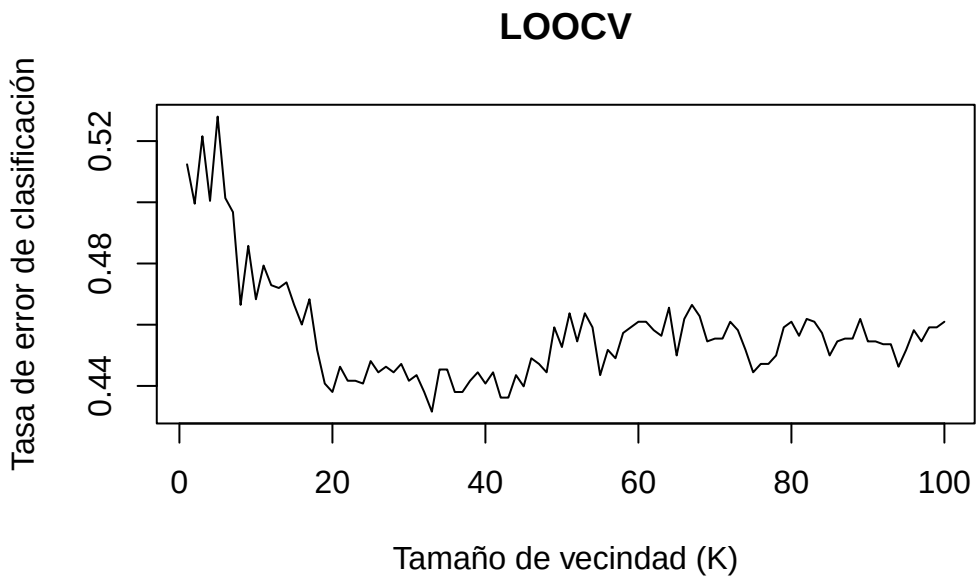
Respecto al uso del modelo **KNN**, no es necesario estandarizar las variables, ya que solo se tiene un predictor, `Lag2`, el cual es numérico, por lo que tampoco se tienen predictores categóricos por binarizar.

Asimismo, usaremos todos los datos de `Weekly` para validación cruzada, en vez de dividir el conjunto de entrenamiento fijado previamente en uno o más pares (nuevo conjunto de entrenamiento, nuevo conjunto de validación). Esto debido a que, con más datos, los modelos se entrenan mejor.

Por último, emplearemos LOOCV y estaremos usando la tasa de error de clasificación, como criterio de selección de los modelos KNN, para distintos valores de K.

```r
# Consideramos el tamaño de la vecindad K entre 1 y 100
numero_casos_K <- 100
error_rate <- rep(0, numero_casos_K)

# Fijamos semilla debido al posible caso de "tie-breaks"
# cuando se vota por la clase por predecir.
set.seed(6174)
for (K in 1:numero_casos_K) {
  error_rate[K] <- mean(
    class::knn.cv(train = Weekly["Lag2"], cl = Weekly$Direction, k = K) != Weekly$Directio
  )
}

plot(
```

```
  error_rate, main = "LOOCV", type = 'l',
  xlab = "Tamaño de vecindad (K)", ylab = "Tasa de error de clasificación"
)
```




```
# Tamaño de vecindad que minimiza la tasa de error de clasificación
optimum_K <- which.min(error_rate)
optimum_K
```

```
[1] 33
```

```
#:::::::::::::::::::::::: #
# Modelo KNN usando K = 33
#:::::::::::::::::::::::: #
# Valores predichos
ypred8 <- class::knn(
  train = training_set["Lag2"], test = validation_set["Lag2"],
  cl = training_set$Direction, k = optimum_K, prob = TRUE
)

# Estimaciones de probabilidades predichas
yprob8 <- attr(ypred8, "prob")

# Matriz de confusión
```

```r
tmp8 <- caret::confusionMatrix(
  data = ypred8, reference = validation_set$Direction,
  positive = "Up", mode = "everything"
)

metricas_modelos <- rbind(
  metricas_modelos,
  data.frame(as.list(c(tmp8[[3]], tmp8[[4]])), row.names = "Modelo 8: KNN")
)
```

```
          Reference
Prediction Up Down
      Up   36   22
      Down 25   21
```

Comparación de los modelos a partir de la pregunta anterior:

```r
knitr::kable(t(metricas_modelos[-c(1, 2),]))
```

	Modelo 3	Modelo 4: probit	Modelo 5: cloglog	Modelo 6: cauchit	Modelo 7: Naive Bayes	Modelo 8: KNN
Accuracy	0.6250000	0.6250000	0.6250000	0.6250000	0.6057692	0.5480769
Kappa	0.1414056	0.1414056	0.1414056	0.1414056	0.0904437	0.0777358
AccuracyLower	0.5246597	0.5246597	0.5246597	0.5246597	0.5051377	0.4474460
AccuracyUpper	0.7180252	0.7180252	0.7180252	0.7180252	0.7002144	0.6458988
AccuracyNull	0.5865385	0.5865385	0.5865385	0.5865385	0.5865385	0.5865385
AccuracyPValue	0.2439500	0.2439500	0.2439500	0.2439500	0.3846868	0.8151603
McnemarPValue	0.0000073	0.0000073	0.0000073	0.0000073	0.0000028	0.7704931
Sensitivity	0.9180328	0.9180328	0.9180328	0.9180328	0.9180328	0.5901639
Specificity	0.2093023	0.2093023	0.2093023	0.2093023	0.1627907	0.4883721
Pos.Pred.Value	0.6222222	0.6222222	0.6222222	0.6222222	0.6086957	0.6206897
Neg.Pred.Value	0.6428571	0.6428571	0.6428571	0.6428571	0.5833333	0.4565217
Precision	0.6222222	0.6222222	0.6222222	0.6222222	0.6086957	0.6206897
Recall	0.9180328	0.9180328	0.9180328	0.9180328	0.9180328	0.5901639
F1	0.7417219	0.7417219	0.7417219	0.7417219	0.7320261	0.6050420
Prevalence	0.5865385	0.5865385	0.5865385	0.5865385	0.5865385	0.5865385
Detection.Rate	0.5384615	0.5384615	0.5384615	0.5384615	0.5384615	0.3461538
Detection.Prevalence	0.8653846	0.8653846	0.8653846	0.8653846	0.8846154	0.5576923
Balanced.Accuracy	0.5636676	0.5636676	0.5636676	0.5636676	0.5404117	0.5392680

- Notamos que los cuatro modelos de tipo `glm()` presentan las mismas métricas respecto al conjunto de validación (accuracy, sensibilidad, etc).
- Los mayores valores de **accuracy** y **coeficiente de Kappa** entre los modelos presentados ocurren para el modelo de regresión logística.
- El modelo **Naive Bayes** presenta la **máxima sensibilidad** y **mínima especificidad**.
- Por último, resaltamos que el modelo **Naive Bayes** no siempre presenta métricas con mayor valor que en el modelo **KNN**. Por ejemplo, **KNN** presenta mayor especificidad que **Naive Bayes**; mas, tiene menor coeficiente de Kappa.

Pregunta g

```
# Curvas ROC
curvas_roc <- list()
curvas_roc[[1]] <- roc(response = validation_set$Direction, predictor = yprob3)
```

```
Setting levels: control = Down, case = Up

Setting direction: controls > cases
```

```
curvas_roc[[2]] <- roc(response = validation_set$Direction, predictor = yprob4)
```

```
Setting levels: control = Down, case = Up
Setting direction: controls > cases
```

```
curvas_roc[[3]] <- roc(response = validation_set$Direction, predictor = yprob5)
```

```
Setting levels: control = Down, case = Up
Setting direction: controls > cases
```

```
curvas_roc[[4]] <- roc(response = validation_set$Direction, predictor = yprob6)
```

```
Setting levels: control = Down, case = Up
Setting direction: controls > cases
```

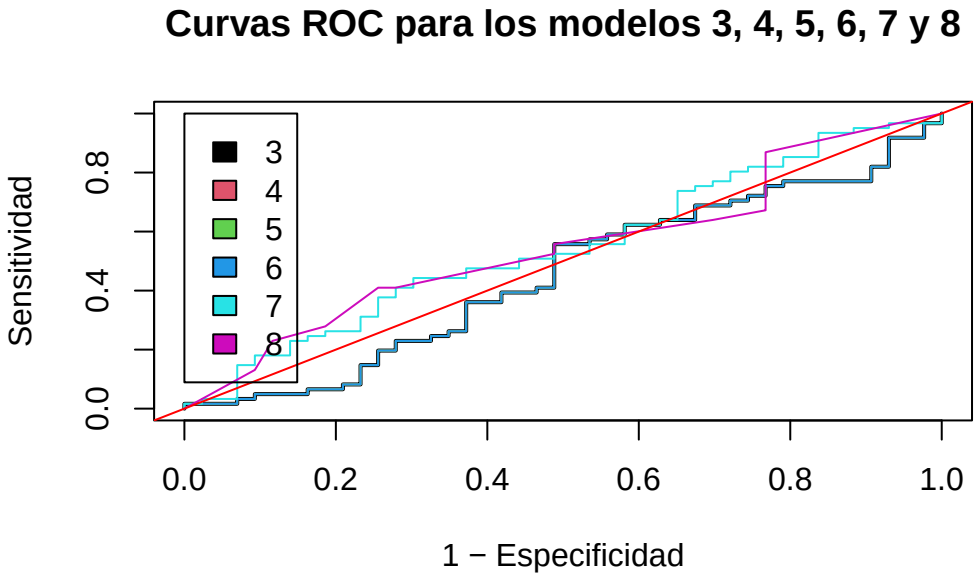
```
curvas_roc[[5]] <- roc(response = validation_set$Direction, predictor = yprob7)
```

```
Setting levels: control = Down, case = Up

Setting direction: controls < cases
```

```
curvas_roc[[6]] <- roc(response = validation_set$Direction, predictor = yprob8)
```

```
Setting levels: control = Down, case = Up
Setting direction: controls < cases
```

```
# Curva ROC
plot(
  1 - curvas_roc[[1]]$specificities, curvas_roc[[1]]$sensitivities,
  ylab = "Sensitividad", xlab = "1 - Especificidad",
  main = "Curvas ROC para los modelos 3, 4, 5, 6, 7 y 8",
  type = "l", col = 1, lwd = 2
)
for (indice in 2:6) {
  lines(1 - curvas_roc[[indice]]$specificities, curvas_roc[[indice]]$sensitivities, col =
}

abline(a = 0, b = 1, col = "red")
legend(0, 1, legend = 3:8, fill = 1:6)
```



De la gráfica notamos que ninguno de los modelos es cercano al modelo ideal, pues las curvas ROC no están cercanas al punto $(0, 1)$ del gráfico.

Asimismo, los modelos asociados al gráfico presentan curvas ROC relativamente cerca a la recta (diagonal) $x = y$. Esto significa que las predicciones generadas por estos modelos no difieren mucho de si se predijese al azar la clase. Esto se evidencia por los valores tan cercanos a cero de los coeficientes de Kappa de estos modelos.

También notamos que la curva ROC asociada al modelo KNN (color morado) es *muy poligonal* (no parece tan *continua* como las otras curvas). Esto es consecuencia de que las estimaciones de probabilidad de pertencia a una clase son mucho menos precisas para KNN, comparado a los otros modelos. Aquello debido a que tal probabilidad se estima según las proporciones de las clases respecto a los elementos de la vecindad; así, para cada vecindad considerada de **K** observaciones.

Por último, inspeccionando una *distancia promedio* entre cada curva ROC y el punto $(0, 1)$, notamos que el modelo **Naive Bayes** (color celeste) podría considerarse el mejor entre los modelos testeados (curva ROC más cercana a $(0, 1)$).

Pregunta h

```
# Particionamos Weekly en 10 partes de
# aproximadamente la misma cantidad de observaciones

# Número de folds
k <- 10

set.seed(1729)
splitPlan <- vtreat::kWayCrossValidation(nRows = nrow(Weekly), k)
```

Para considerar distintos puntos de corte, fijaremos la métrica área bajo la curva ROC (**AUC**) como criterio de comparación de modelos.

```
resultados_finales <- data.frame(
  Modelo = c(
    "Binomial, Link logit", "Binomial, Link probit",
    "Binomial, Link cloglog", "Binomial, Link cauchit",
    "Naive Bayes", "KNN, K = 33"
  )
)

# Para cada uno de los 10 folds, guardamos el valor hallado de AUC
```

```r
for (fold in 1:k) {
  resultados_finales[paste0("AUC_", fold)] <- 0
}
resultados_finales
```

```
                 Modelo AUC_1 AUC_2 AUC_3 AUC_4 AUC_5 AUC_6 AUC_7 AUC_8 AUC_9
1   Binomial, Link logit     0     0     0     0     0     0     0     0     0
2  Binomial, Link probit     0     0     0     0     0     0     0     0     0
3 Binomial, Link cloglog     0     0     0     0     0     0     0     0     0
4 Binomial, Link cauchit     0     0     0     0     0     0     0     0     0
5            Naive Bayes     0     0     0     0     0     0     0     0     0
6            KNN, K = 33     0     0     0     0     0     0     0     0     0
  AUC_10
1      0
2      0
3      0
4      0
5      0
6      0
```

```r
for(fold in 1:k) {
  split <- splitPlan[[fold]]

  # Hallamos las probabilidades predichas para el
  # conjunto de validación, para cada uno de los seis modelos .
  for (fila in 1:6) {
    prob_pred <- NULL

    # Modelo glm(..., binomial("logit"))
    if (fila == 1) {
      modelo <- glm(
        Direction ~ Lag2, family = binomial(link = "logit"),
        data = Weekly[split$train,]
      )
      prob_pred <- predict(
        modelo, newdata = Weekly[split$app,], type = 'response'
      )
    }

    # Modelo glm(..., binomial("probit"))
    if (fila == 2) {
```

```
  modelo <- glm(
    Direction ~ Lag2, family = binomial(link = "probit"),
    data = Weekly[split$train,]
  )
  prob_pred <- predict(
    modelo, newdata = Weekly[split$app,], type = 'response'
  )
}

# Modelo glm(..., binomial("cloglog"))
if (fila == 3) {
  modelo <- glm(
    Direction ~ Lag2, family = binomial(link = "cloglog"),
    data = Weekly[split$train,]
  )
  prob_pred <- predict(
    modelo, newdata = Weekly[split$app,], type = 'response'
  )
}

# Modelo glm(..., binomial("cauchit"))
if (fila == 4) {
  modelo <- glm(
    Direction ~ Lag2, family = binomial(link = "cauchit"),
    data = Weekly[split$train,]
  )
  prob_pred <- predict(
    modelo, newdata = Weekly[split$app,], type = 'response'
  )
}

# Modelo Naive Bayes
if (fila == 5) {
  modelo <- naive_bayes(
    x = Weekly[split$train,]["Lag2"], y = Weekly[split$train,]$Direction,
    usekernel = TRUE
  )
  prob_pred <- predict(
    modelo, newdata = Weekly[split$app,]["Lag2"],  type = 'prob'
  )[,2]
}
```

```
    # Modelo KNN con K = 33
    if (fila == 6) {
      modelo <- class::knn(
        train = Weekly[split$train,]["Lag2"], test = Weekly[split$app,]["Lag2"],
        cl = Weekly[split$train,]$Direction, k = optimum_K, prob = TRUE
      )
      prob_pred <- attr(modelo, "prob")
    }


    # Grabar el valor de AUC asociado al fold, para cada modelo .
    analysis <- roc(
      response = Weekly[split$app,]$Direction, predictor = prob_pred
    )
    resultados_finales[fila, paste0("AUC_", fold)] <- analysis$auc
  }
}
```

```
Setting levels: control = Down, case = Up

Setting direction: controls < cases

Setting levels: control = Down, case = Up

Setting direction: controls < cases

Setting levels: control = Down, case = Up

Setting direction: controls < cases

Setting levels: control = Down, case = Up

Setting direction: controls < cases

Setting levels: control = Down, case = Up

Setting direction: controls > cases
```

```
Setting levels: control = Down, case = Up

Setting direction: controls < cases

Setting levels: control = Down, case = Up

Setting direction: controls > cases

Setting levels: control = Down, case = Up

Setting direction: controls > cases

Setting levels: control = Down, case = Up

Setting direction: controls > cases

Setting levels: control = Down, case = Up

Setting direction: controls > cases

Setting levels: control = Down, case = Up

Setting direction: controls < cases

Setting levels: control = Down, case = Up

Setting direction: controls < cases

Setting levels: control = Down, case = Up

Setting direction: controls > cases

Setting levels: control = Down, case = Up

Setting direction: controls > cases

Setting levels: control = Down, case = Up
```

```
Setting direction: controls > cases

Setting levels: control = Down, case = Up

Setting direction: controls > cases

Setting levels: control = Down, case = Up

Setting direction: controls < cases

Setting levels: control = Down, case = Up

Setting direction: controls < cases

Setting levels: control = Down, case = Up

Setting direction: controls < cases

Setting levels: control = Down, case = Up

Setting direction: controls < cases

Setting levels: control = Down, case = Up

Setting direction: controls < cases

Setting levels: control = Down, case = Up

Setting direction: controls < cases

Setting levels: control = Down, case = Up

Setting direction: controls < cases

Setting levels: control = Down, case = Up

Setting direction: controls < cases
```

```
Setting levels: control = Down, case = Up

Setting direction: controls < cases

Setting levels: control = Down, case = Up

Setting direction: controls < cases

Setting levels: control = Down, case = Up

Setting direction: controls < cases

Setting levels: control = Down, case = Up

Setting direction: controls < cases

Setting levels: control = Down, case = Up

Setting direction: controls < cases

Setting levels: control = Down, case = Up

Setting direction: controls < cases

Setting levels: control = Down, case = Up

Setting direction: controls > cases

Setting levels: control = Down, case = Up

Setting direction: controls > cases

Setting levels: control = Down, case = Up

Setting direction: controls > cases

Setting levels: control = Down, case = Up
```

```
Setting direction: controls > cases

Setting levels: control = Down, case = Up

Setting direction: controls > cases

Setting levels: control = Down, case = Up

Setting direction: controls < cases

Setting levels: control = Down, case = Up

Setting direction: controls < cases

Setting levels: control = Down, case = Up

Setting direction: controls < cases

Setting levels: control = Down, case = Up

Setting direction: controls < cases

Setting levels: control = Down, case = Up

Setting direction: controls < cases

Setting levels: control = Down, case = Up

Setting direction: controls > cases

Setting levels: control = Down, case = Up

Setting direction: controls < cases

Setting levels: control = Down, case = Up

Setting direction: controls < cases
```

```
Setting levels: control = Down, case = Up

Setting direction: controls < cases

Setting levels: control = Down, case = Up

Setting direction: controls < cases

Setting levels: control = Down, case = Up

Setting direction: controls < cases

Setting levels: control = Down, case = Up

Setting direction: controls > cases

Setting levels: control = Down, case = Up

Setting direction: controls < cases

Setting levels: control = Down, case = Up

Setting direction: controls > cases

Setting levels: control = Down, case = Up

Setting direction: controls > cases

Setting levels: control = Down, case = Up

Setting direction: controls > cases

Setting levels: control = Down, case = Up

Setting direction: controls > cases

Setting levels: control = Down, case = Up
```

```
Setting direction: controls > cases

Setting levels: control = Down, case = Up

Setting direction: controls < cases

Setting levels: control = Down, case = Up

Setting direction: controls > cases

Setting levels: control = Down, case = Up

Setting direction: controls > cases

Setting levels: control = Down, case = Up

Setting direction: controls > cases

Setting levels: control = Down, case = Up

Setting direction: controls > cases

Setting levels: control = Down, case = Up

Setting direction: controls > cases

Setting levels: control = Down, case = Up

Setting direction: controls < cases
```

Modelo	AUC_1	AUC_2	AUC_3	AUC_4	AUC_5	AUC_6	AUC_7	AUC_8	AUC_9	AUC_10

```
knitr::kable(resultados_finales)
```

Modelo	AUC_1	AUC_2	AUC_3	AUC_4	AUC_5	AUC_6	AUC_7	AUC_8	AUC_9	AUC_10
Binomial, Link logit	0.6188187	0.7523050	0.8487562	0.7256503	0.9608606	0.6511447	0.8574637	0.7509684	0.4530737	0.7473630
Binomial, Link probit	0.6188187	0.7523050	0.8487562	0.7256503	0.9608606	0.6511447	0.8574637	0.7509684	0.4530737	0.7473630
Binomial, Link cloglog	0.6188187	0.7523050	0.8487562	0.7256503	0.9608606	0.6511447	0.8574637	0.7509684	0.4530737	0.7473630
Binomial, Link cauchit	0.6188187	0.7523050	0.8487562	0.7256503	0.9608606	0.6511447	0.8574637	0.7509684	0.4530737	0.7473630
Naive Bayes	0.4938187	0.7563728	0.8563610	0.6576012	0.9559426	0.6253097	0.8466304	0.6550717	0.3652971	0.6599053
KNN, K = 33	0.5453297	0.7565932	0.8561833	0.7501372	0.7501707	0.7537373	0.7512137	0.7534253	0.3950614	0.7557926

```
# Promediamos de manera ponderada los diez valores de AUC,
# respecto a la cantidad de observaciones en los 10 conjuntos de validación
resultados_finales["AUC_ponderado"] <- 0
sum_sizes_validation_sets <- 0

# Suma ponderada
for (fila in 1:6) {
  for (fold in 1:k) {
    resultados_finales[fila, "AUC_ponderado"] <- resultados_finales[fila, "AUC_ponderado"]

    if (fila == 1) {
      sum_sizes_validation_sets <- sum_sizes_validation_sets + length(splitPlan[[fold]]$ap
    }
  }
}

# Dividir entre la suma de los tamaños de los 10 conjuntos de validación
resultados_finales["AUC_ponderado"] <- resultados_finales["AUC_ponderado"] / sum_sizes_val
```

```
knitr::kable(resultados_finales[, c("Modelo", "AUC_ponderado")])
```

Modelo	AUC_ponderado
Binomial, Link logit	0.5402487
Binomial, Link probit	0.5402487
Binomial, Link cloglog	0.5402487
Binomial, Link cauchit	0.5402487
Naive Bayes	0.5587152
KNN, K = 33	0.5345266

Como modelos de clasificación con mayor **AUC** suelen ser mejores, de la tabla previa concluimos que el modelo **Naive Bayes** tuvo el mejor desempeño entre los seis modelos, seguido por los cuatro modelos de tipo **glm(...)**; y, por último, **KNN**, modelo de menor desempeño.

Estos resultados son consistentes con lo obtenido en la pregunta **g**, puesto que los modelos cuyas curvas ROC estaban más cercanas al punto $(0, 1)$ han sido los mismos que, en esta pregunta, hemos verificado tienen mejor desempeño predictivo.

Pregunta 2

CV incorrecto

```
library(boot)
```

```
Warning: package 'boot' was built under R version 4.1.3
```

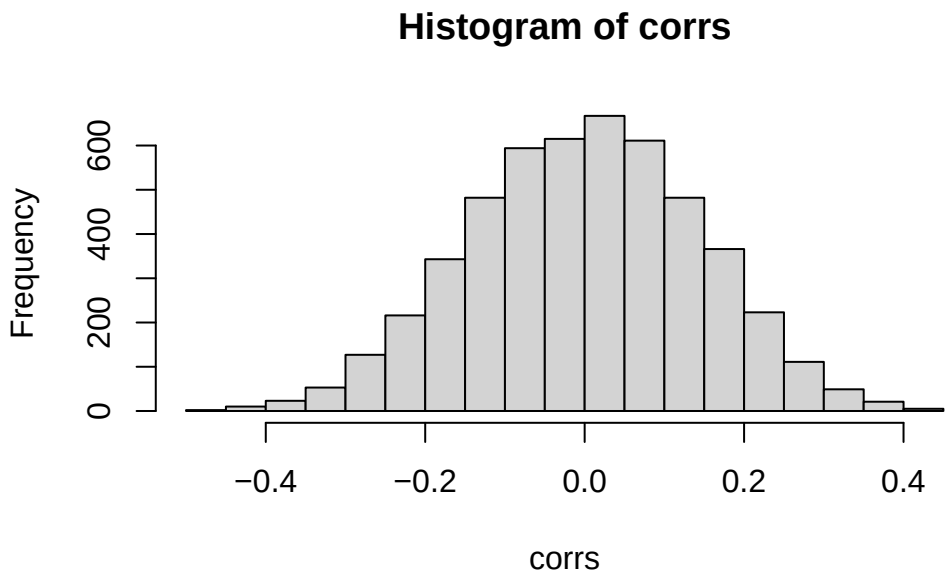
```
# GENERAR DATOS; utilizar una semilla para la reproducibilidad
set.seed(4268)
n = 50 # número de observaciones
p = 5000 # número de predictores

d = 25 # principales predictores correlacionados elegidos

# generando datos para los predictores
xs = matrix(rnorm(n * p, 0, 4), ncol = p, nrow = n) # forma simple de predictores no corre

dim(xs) # n times p
```

```
[1]   50 5000
```

```r
# generar etiquetas de clase independientes de los predictores, por
# lo que si todo se clasifica como clase 1, esperamos un 50% de
# errores en general
ys = c(rep(0, n/2), rep(1, n/2)) # Ahora realmente el 50% de cada una
table(ys)
```

```
ys
 0  1
25 25
```

```r
corrs = apply(xs, 2, cor, y = ys)
hist(corrs)
```




```r
selected = order(corrs^2, decreasing = TRUE)[1:d] # top d correlaciones seleccionadas
data = data.frame(ys, xs[, selected])
```

```r
logfit = glm(ys ~ ., family = "binomial", data = data)
```

```
Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
cost <- function(r, pi = 0) mean(abs(r - pi) > 0.5)
kfold = 10
cvres = cv.glm(data = data, cost = cost, glmfit = logfit, K = kfold)
```

```
Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
cvres$delta
```

```
[1] 0 0
```

CV correcto

```r
# Fijamos índices que representan un orden aleatorio
# de las 50 observaciones (sin repetición)
reorder = sample(1:n, replace = FALSE)

validclass = NULL

# Para cada uno de los 10 folds ()
for (i in 1:kfold) {
  # Cantidad (aproximada) de observaciones por fold
```

```r
neach = n/kfold # (vale 5 en este ejemplo)

# Fijamos los identificadores del conjunto de entrenamiento.
# Para el fold 1:
#   trainids es todos los ids en el dataset completo (1 a 50),
#   salvo los ids de 1 a 5.
# Para el fold 2:
#   trainids es todos los ids en el dataset completo (1 a 50),
#   salvo los ids de 6 a 11.
# De esa manera, de 5 en 5, por un total de 10 veces, separamos
# los datos en conjuntos de entrenamiento y de validación
trainids = setdiff(1:n, (((i - 1) * neach + 1):(i * neach)))

# Los ids de trainids no se usarán respecto al número de fila en
# el conjunto de datos completo (xs), sino respecto al reordenamiento
# aleatorio que se fijó previamente en la variable reorder.
# Así, separamos los predictores y la response, creando, para este fold,
# el conjunto de entrenamiento (traindata) y el conjunto de validación (validdata)
traindata = data.frame(xs[reorder[trainids], ], ys[reorder[trainids]])
validdata = data.frame(xs[reorder[-trainids], ], ys[reorder[-trainids]])

# Asignamos nombres a las columnas del data frame para poder usar
# fácilmente la fórmula en la función glm()
colnames(traindata) = colnames(validdata) = c(paste("X", 1:p), "y")

# Para cada uno de los predictores, hallamos su correlación con
# la variable respuesta, respecto a los datos del training dataset .
foldcorrs = apply(traindata[, 1:p], 2, cor, y = traindata[, p + 1])

# Seleccionamos los d (25) predictores mayor correlacionados con la response
selected = order(foldcorrs^2, decreasing = TRUE)[1:d] #top d correlaciones seleccionadas

# Consideramos solo aquellos d (predictores) para el modelo por entrenar,
# así que filtramos la data de entrenamiento previa.
data = traindata[, c(selected, p + 1)]
trainlogfit = glm(y ~ ., family = binomial, data = data)

# Predecimos la probabilidad que asignada el modelo entrenado,
# para los datos de validación (filtrando también ahí solo las d
# variables de mayor correlación con la variable response)
pred = plogis(predict.glm(trainlogfit, newdata = validdata[, selected]))
```

```
  # Asignamos la clase 1 si la probabilidad de pertenencia predicha es mayor a 0.5
  validclass = c(validclass, ifelse(pred > 0.5, 1, 0))
}
```

```
Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
# Matriz de confusión asociada al modelo
table(ys[reorder], validclass)
```

```
   validclass
     0  1
  0  9 16
  1 12 13
```

```
# Porcentaje (*100) de predicciones incorrectas .
1 - sum(diag(table(ys[reorder], validclass)))/n
```

```
[1] 0.56
```