Tarea 3

Aprendizaje Supervisado

González Feria Juan Rosendo
González González Elvira
Miranda Peñafiel Melissa S.
Nicolas Mata Jesús
Pacheco Martínez Mariana

Ejercicio 1

Considere la base de datos `rbind(Pima.tr, Pima.te)` del paquete `MASS`.

```
## 'data.frame':    532 obs. of  8 variables:
##  $ npreg: int  5 7 5 0 0 5 3 1 3 2 ...
##  $ glu  : int  86 195 77 165 107 97 83 193 142 128 ...
##  $ bp   : int  68 70 82 76 60 76 58 50 80 78 ...
##  $ skin : int  28 33 41 43 25 27 31 16 15 37 ...
##  $ bmi  : num  30.2 25.1 35.8 47.9 26.4 35.6 34.3 25.9 32.4 43.3 ...
##  $ ped  : num  0.364 0.163 0.156 0.259 0.133 ...
##  $ age  : int  24 55 35 26 23 52 25 24 63 31 ...
##  $ type : Factor w/ 2 levels "No","Yes": 1 2 1 1 1 2 1 1 1 2 ...
```

Considere el ajuste de los modelos o método siguientes:

a) Análisis de discriminante lineal
b) *Naive Bayes*
c) Regresión Logística
d) *SVM*

En este caso es posible utilizar los cuatro métodos como clasificadores al considerar que las variables explicativas o predictores son todas de tipo continuo. Es un problema de aprendizaje supervisado binario, la variable `type` determina los dos grupos o clases.

Para cada modelo o método considere optimizar su capacidad predictiva. Esto lo puede hacer al considerar interacciones entre variables o transformación de variables e.g. elevarlas a alguna potencia. Uso de selección de variables con el uso de la función `step()`. Distintos tipos y valores de `(kernel,cost,gamma)`.

Finalmente, reporte las especificaciones de cada uno de los cuatro modelos o método seleccionado, uno para cada inciso. Reporte las tasas de clasificación errónea por grupo y global, aparentes, y no aparentes obtenidas por el método de repetición training/test. Hágalo de forma tabular.

Comente brevemente sobre los valores obtenidos por cada modelo.

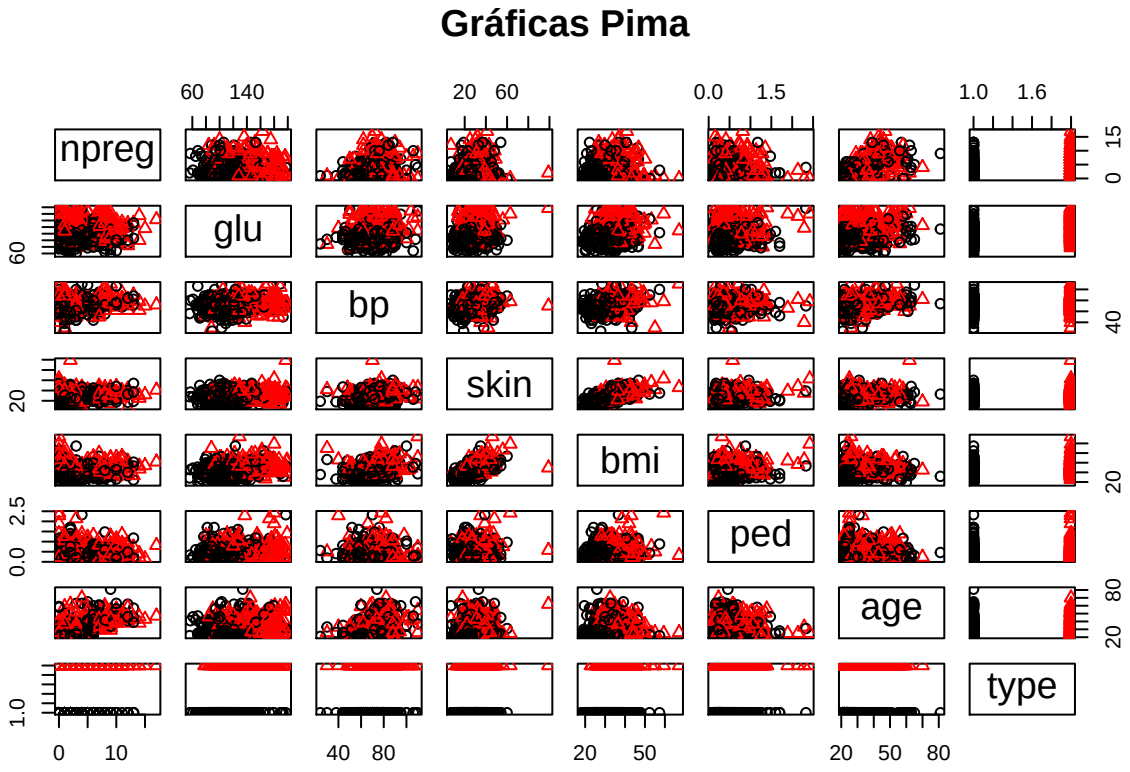
Contexto de la base. La base de datos recopila ciertos datos de mujeres mayores de 21 años, estos datos tienen la peculiaridad de que pueden ser desencadenantes de diabetes según la Organización Mundial de la Salud. Algunos de los parámetros de esta base son el número de embarazos, la concentración de glucosa en la sangre, el índice de masa corporal, etc. La variable `type` es una variable binaria que indica si el individuo tienen o no diabetes según los parámetros de la OMS.

Antes de empezar haciendo los métodos analicemos un poco la base de datos. Veamos primero la matriz de correlaciones entre variables [1].

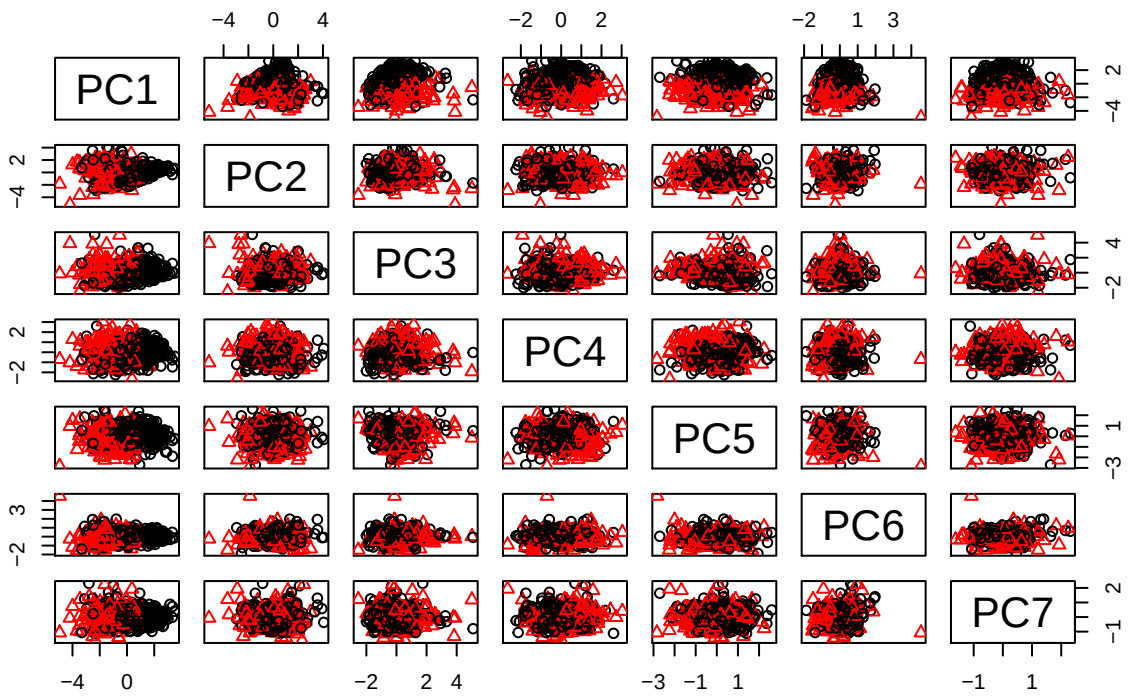
[1] Para esto hemos quitado la variable categórica y hemos convertido todas las variables restantes a tipo numérico.

```
##       npreg   glu    bp  skin   bmi   ped   age
## npreg 1.000 0.125 0.205 0.095 0.009 0.007 0.641
## glu   0.125 1.000 0.219 0.227 0.247 0.166 0.279
## bp    0.205 0.219 1.000 0.226 0.307 0.008 0.347
## skin  0.095 0.227 0.226 1.000 0.647 0.119 0.161
## bmi   0.009 0.247 0.307 0.647 1.000 0.151 0.073
## ped   0.007 0.166 0.008 0.119 0.151 1.000 0.072
## age   0.641 0.279 0.347 0.161 0.073 0.072 1.000
```

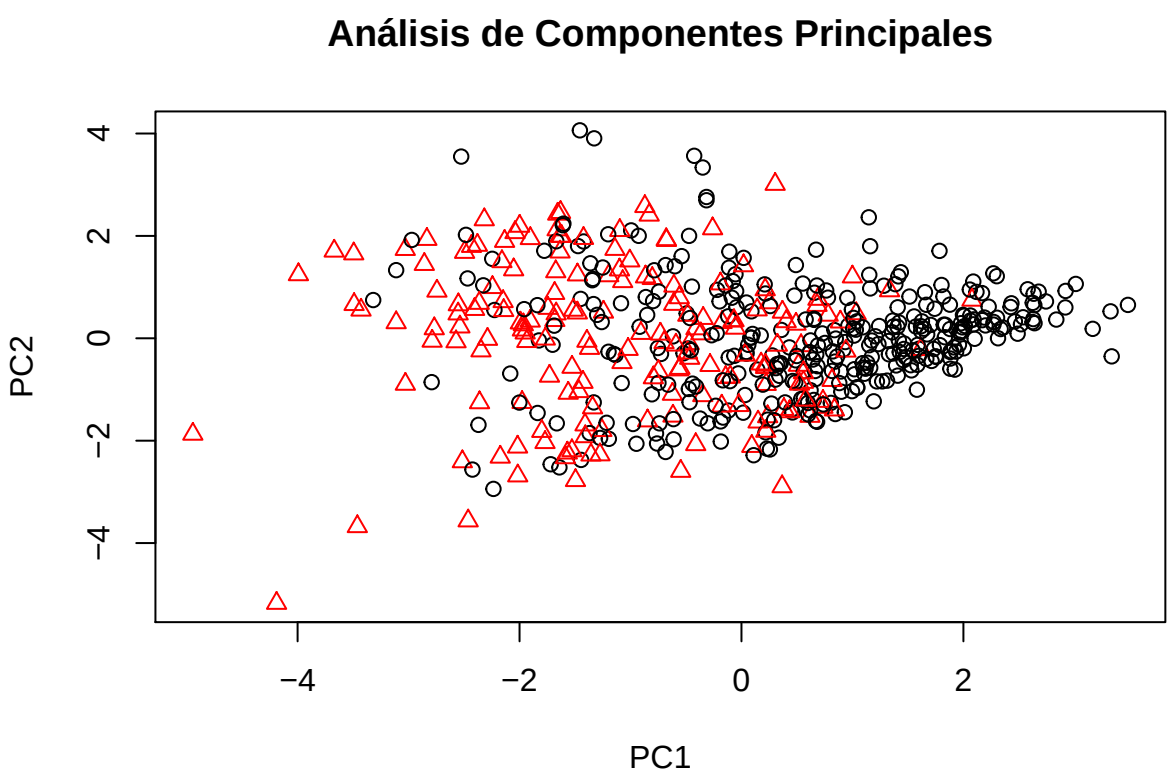
No parece haber una correlación tan alta entre variables, siendo la correlación entre `bmi` y `skin` la más alta (0.647). Veamos como se ven las variables graficadas por pares:



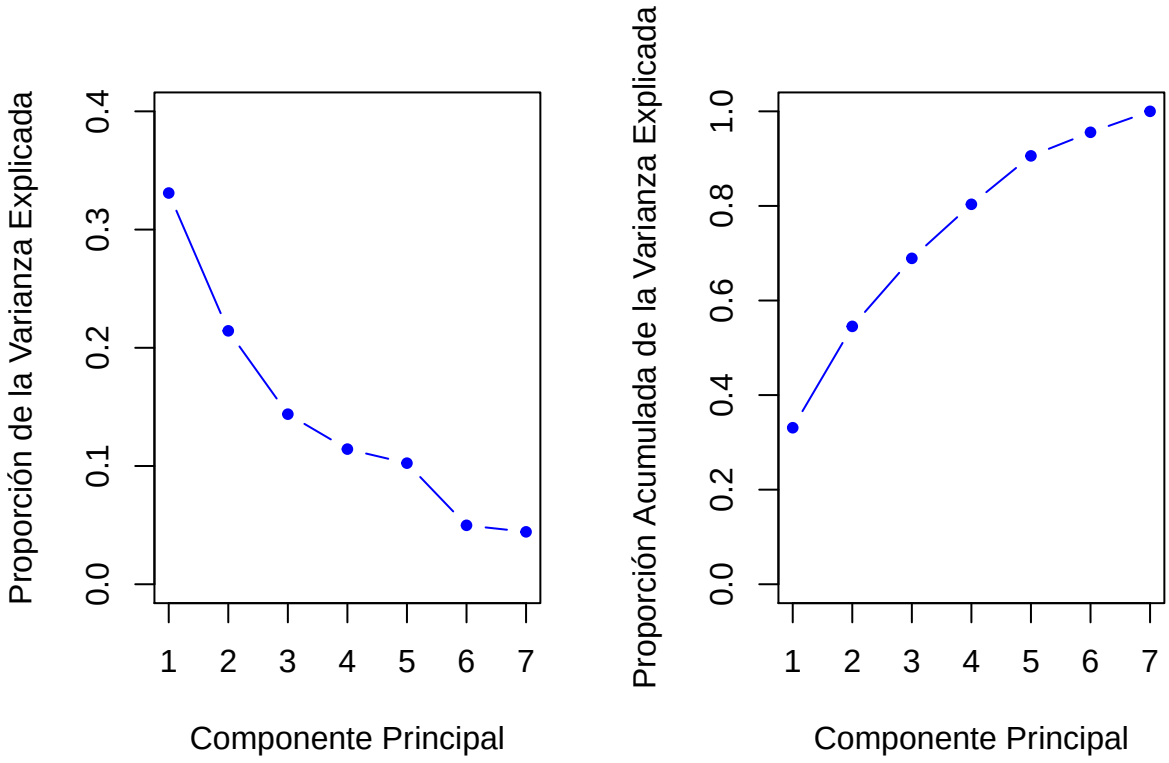
Nuevamente no hay alguna información relevante que nos pueda ayudar a clasificar usando los datos como ejes para graficar. Veamos mejor una gráfica de las componentes principales:



Se puede apreciar mejor como se separan los grupos. Veamos la gráfica de las primeras dos componentes principales:



Finalmente veamos las proporciones de la varianza que son explicadas por cada componente principal:



Se puede ver que dos componentes explican menos del 10% de la varianza de los datos. En otro caso podría ayudar para reducir dimensiones, pero dado que aún el número de dimensiones es bajo y no supone un costo computacional considerable, entonces podemos tomar todas las variables y trabajar con ellas.

Ya con una idea dada por la visualización de los datos podemos empezar a construir nuestros modelos.

a) Análisis de Discriminante Lineal.

Para este método proponemos cinco modelos distintos:

1) El primero tiene a todos los predictores como variables explicativas.
2) El segundo tiene todas las interacciones hasta segundo grado.
3) El tercero vuelve a ser lineal, pero ya no tiene a las variables `bp, skin` y `age`.[2]
4) Para el cuarto se tiene un modelo con interacciones hasta tercer orden.
5) Finalmente se tiene un modelo con interacciones de hasta cuarto orden. [3]

Para mejor manejo todos los modelos se guardaron en una lista y para ellos se tiene su matriz de confusión con sus respectivas tasas globales y por grupo.[4]

[2]La elección no es al azar, resulta que, a primera vista, en el modelo de regresión logística estas variables parecen no ser estadísticamente significativas.

[3]Originalmente este no se tenía, pero parecía que había cierta tendencia a que las tazas fueran más pequeñas conforme se incrementaban las interacciones

[4]Por comodidad no se muestra el código sino solamente los resultados. El código se anexa al final

```
##      Predicho
## Real   No Yes
##   No  317  38
##   Yes  75 102
```

```
##            No    Yes
## 0.2124 0.1070 0.4237
```

```
##      Predicho
## Real   No Yes
##   No  318  37
##   Yes  70 107
```

```
##            No    Yes
## 0.2011 0.1042 0.3955
```

```
##      Predicho
## Real   No Yes
##   No  317  38
##   Yes  74 103
```

```
##            No    Yes
## 0.2105 0.1070 0.4181
```

```
##      Predicho
## Real   No Yes
##   No  316  39
##   Yes  60 117
```

```
##            No    Yes
## 0.1861 0.1099 0.3390
```

```
##      Predicho
## Real   No Yes
##   No  317  38
##   Yes  75 102
```

```
##            No    Yes
## 0.2124 0.1070 0.4237
```

Parece ser que el modelo con interacciones de hasta tercer orden tiene las tazas más bajas con respecto a los demás modelos considerados.

b) *Naive Bayes.*

Para este método suena lógico que tomemos los mismos cinco modelos que en el método anterior, sin embargo el método de Naive Bayes *no permite interacciones* [5]por lo que solamente hemos considerado el modelo simple y el modelo sin las variables `bp, skin` y `age`. Análogamente se muestran las matrices de confusión y las tazas de error:

```
##      Predicho
## Real   No Yes
##   No  289  66
##   Yes  61 116
```

```
##            No    Yes
## 0.2387 0.1859 0.3446
```

[5]El hecho de que no se puedan interacciones se debe a las hipótesis que sustentan el modelo, pues se supone que las variables son independientes.

```
##      Predicho
## Real   No Yes
##   No  313  42
##   Yes  70 107
```

```
##            No    Yes
## 0.2105 0.1183 0.3955
```

Podemos observar que en el segundo modelo, sin las variables `bp, skin` y `age`, la tasa de error global y en el grupo No es menor que en el modelo con todas las variables. Siendo así elegiremos ese para comparar con el resto.

c) Regresión Logística.

Ajustemos nuestros datos ahora usando una regresión logística, empecemos con un modelo con todas las variables:

```
##
## Call:
## glm(formula = type ~ ., family = binomial(link = "logit"), data = Pima)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -3.010  -0.661  -0.369   0.643   2.479
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -9.55465    0.99422   -9.61  < 2e-16 ***
## npreg        0.12252    0.04374    2.80  0.00510 **
## glu          0.03532    0.00424    8.32  < 2e-16 ***
## bp          -0.00770    0.01031   -0.75  0.45560
## skin         0.00677    0.01476    0.46  0.64624
## bmi          0.08268    0.02333    3.54  0.00040 ***
## ped          1.30871    0.36404    3.59  0.00032 ***
## age          0.02637    0.01400    1.88  0.05958 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 676.79  on 531  degrees of freedom
## Residual deviance: 466.32  on 524  degrees of freedom
## AIC: 482.3
##
## Number of Fisher Scoring iterations: 5
```

```
## Single term deletions
##
## Model:
## type ~ npreg + glu + bp + skin + bmi + ped + age
##        Df Deviance AIC  LRT Pr(>Chi)
## <none>         466 482
## npreg   1      474 488  8.1  0.00448 **
## glu     1      553 567 86.6  < 2e-16 ***
## bp      1      467 481  0.6  0.45624
## skin    1      467 481  0.2  0.64536
## bmi     1      479 493 13.1  0.00030 ***
```

```
## ped     1      480 494 13.6  0.00023 ***
## age     1      470 484  3.6  0.05839 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Podemos ver que las variables `bp, skin` y `age` parecen no ser estadísticamente significativas, para ver si la correlación entre las otras variables no influye separamos las variables y hacemos otras tres regresiones logísticas:

```
reglog_Pima11 <- glm(type~bp, data = Pima, family = "binomial"(link = "logit"))
summary(reglog_Pima11)
```

```
##
## Call:
## glm(formula = type ~ bp, family = binomial(link = "logit"), data = Pima)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.417  -0.920  -0.779   1.312   2.101
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -3.07945    0.58755   -5.24  1.6e-07 ***
## bp           0.03297    0.00794    4.15  3.3e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 676.79  on 531  degrees of freedom
## Residual deviance: 658.52  on 530  degrees of freedom
## AIC: 662.5
##
## Number of Fisher Scoring iterations: 4
```

Parece ser que la variable `bp` *Si* es estadísticamente significativa. Veamos ahora a la variables `skin`:

```
reglog_Pima12 <- glm(type~skin, data = Pima, family = "binomial"(link = "logit"))
summary(reglog_Pima12)
```

```
##
## Call:
## glm(formula = type ~ skin, family = binomial(link = "logit"),
##     data = Pima)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.599  -0.899  -0.694   1.264   2.046
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.34478    0.31483   -7.45  9.5e-14 ***
## skin         0.05493    0.00977    5.62  1.9e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 676.79  on 531  degrees of freedom
## Residual deviance: 641.36  on 530  degrees of freedom
## AIC: 645.4
##
## Number of Fisher Scoring iterations: 4
```

La variable `skin` también es estadísticamente significativa. Veamos finalmente la variable `age`:

```
reglog_Pima13 <- glm(type~age, data = Pima, family = "binomial"(link = "logit"))
summary(reglog_Pima13)
```

```
##
## Call:
## glm(formula = type ~ age, family = binomial(link = "logit"),
##     data = Pima)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -2.203  -0.802  -0.681   1.146   1.803
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.71482    0.31471   -8.63  < 2e-16 ***
## age          0.06234    0.00915    6.82  9.3e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 676.79  on 531  degrees of freedom
## Residual deviance: 625.09  on 530  degrees of freedom
## AIC: 629.1
##
## Number of Fisher Scoring iterations: 4
```

Concluimos con esto que *todas las variables son estadísticamente significativas*, por lo que el hecho de que no parezcan estadísticamente significativas puede deberse a las correlaciones que existen entre variables.

En el modelo de regresión logística existe una forma de encontrar el modelo óptimo y es usando la función `step`. En este caso tomaremos los criterios AIC y BIC para determinar el mejor modelo. Además del modelo lineal usaremos un modelo con interacciones de hasta segundo grado y otro con interacciones de hasta tercer orden[6]:

Para el primer modelo, en el que usamos todas las variables se tienen las tasas de error siguientes:

```
##            No    Yes
## 0.2124 0.1070 0.4237
```

```
## Single term deletions
##
## Model:
## type ~ npreg + glu + bp + skin + bmi + ped + age
##        Df Deviance AIC  LRT Pr(>Chi)
## <none>         466 482
```

[6]Nuevamente omitimos el código para no hacer tan grande el documento. El código completo se anexa al final

```
## npreg   1      474 488  8.1  0.00448 **
## glu     1      553 567 86.6  < 2e-16 ***
## bp      1      467 481  0.6  0.45624
## skin    1      467 481  0.2  0.64536
## bmi     1      479 493 13.1  0.00030 ***
## ped     1      480 494 13.6  0.00023 ***
## age     1      470 484  3.6  0.05839 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

El segundo modelo es optimizado usando la función `step` y el criterio AIC, por lo que se tienen las siguientes variables que son estadísticamente significativas, el valor de AIC y las tasas de error:

```
##            No    Yes
## 0.2030 0.1014 0.4068
```

```
## Single term deletions
##
## Model:
## type ~ npreg + glu + bmi + ped + age
##        Df Deviance AIC  LRT Pr(>Chi)
## <none>         467 479
## npreg   1      475 485  8.4  0.00385 **
## glu     1      554 564 86.6  < 2e-16 ***
## bmi     1      491 501 24.1    9e-07 ***
## ped     1      481 491 14.0  0.00018 ***
## age     1      470 480  3.2  0.07284 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

El tercer modelo es análogo al segundo, solo que ahora es el criterio BIC el que se elige para optimizar el modelo. Se tiene entonces las siguientes tasas de error:

```
##              No     Yes
## 0.20113 0.09577 0.41243
```

```
## Single term deletions
##
## Model:
## type ~ npreg + glu + bmi + ped
##        Df Deviance AIC  LRT Pr(>Chi)
## <none>         470 480
## npreg   1      497 505 26.7  2.4e-07 ***
## glu     1      569 577 99.1  < 2e-16 ***
## bmi     1      494 502 23.7  1.1e-06 ***
## ped     1      485 493 14.9  0.00011 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

El cuarto modelo es una optimización usando el criterio AIC nuevamente y escogiendo las variables que sean estadísticamente significativas, pero ahora las variables tienen interacciones de hasta segundo orden:

```
##            No    Yes
## 0.2049 0.1211 0.3729
```

```
## Single term deletions
##
## Model:
```

```
## type ~ npreg + glu + skin + bmi + ped + age + npreg:glu + glu:skin +
##     glu:bmi + glu:ped + skin:age + bmi:age
##           Df Deviance AIC  LRT Pr(>Chi)
## <none>            452 478
## npreg:glu  1      455 479 2.54   0.1108
## glu:skin   1      454 478 2.27   0.1322
## glu:bmi    1      455 479 2.99   0.0839 .
## glu:ped    1      460 484 7.49   0.0062 **
## skin:age   1      456 480 4.18   0.0409 *
## bmi:age    1      455 479 2.95   0.0860 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

El quinto modelo es análogo al cuarto, pero ahora usando el criterio BIC:

```
##            No    Yes
## 0.2105 0.1155 0.4011
```

```
## Single term deletions
##
## Model:
## type ~ npreg + glu + bmi + ped + glu:ped
##         Df Deviance AIC   LRT Pr(>Chi)
## <none>          463 475
## npreg    1      491 501 27.59  1.5e-07 ***
## bmi      1      489 499 25.79  3.8e-07 ***
## glu:ped  1      470 480  7.03    0.008 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

El sexto modelo modelo es optimizado usando el criterio AIC y con variables que pueden incluir interacciones de hasta tercer orden

```
##            No    Yes
## 0.1823 0.1099 0.3277
```

```
## Single term deletions
##
## Model:
## type ~ npreg + glu + bp + skin + bmi + ped + age + npreg:glu +
##     npreg:bp + npreg:skin + npreg:bmi + npreg:ped + npreg:age +
##     glu:ped + bp:skin + bp:bmi + bp:ped + bp:age + skin:ped +
##     skin:age + bmi:ped + bmi:age + ped:age + npreg:glu:ped +
##     npreg:bp:skin + npreg:bp:bmi + npreg:bp:ped + npreg:bmi:ped +
##     npreg:bmi:age + bp:skin:ped + bp:skin:age + bp:bmi:ped +
##     bp:bmi:age + bp:ped:age + bmi:ped:age
##                Df Deviance AIC   LRT Pr(>Chi)
## <none>                407 479
## npreg:glu:ped  1      411 481  3.83   0.0503 .
## npreg:bp:skin  1      414 484  7.13   0.0076 **
## npreg:bp:bmi   1      411 481  4.03   0.0446 *
## npreg:bp:ped   1      410 480  2.21   0.1374
## npreg:bmi:ped  1      410 480  3.02   0.0822 .
## npreg:bmi:age  1      416 486  8.52   0.0035 **
## bp:skin:ped    1      412 482  4.68   0.0305 *
## bp:skin:age    1      411 481  3.82   0.0506 .
## bp:bmi:ped     1      422 492 15.20  9.7e-05 ***
```

```
## bp:bmi:age     1      413 483  5.48   0.0192 *
## bp:ped:age     1      416 486  8.62   0.0033 **
## bmi:ped:age    1      410 480  3.07   0.0800 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Finalmente el séptimo modelo es análogo al sexto, pero ahora el criterio de decisión es el BIC:

```
##           No    Yes
## 0.2068 0.1296 0.3616
```

```
## Single term deletions
##
## Model:
## type ~ npreg + glu + bp + bmi + ped + age + npreg:bmi + npreg:age +
##     glu:ped + bp:bmi + bp:ped + bmi:ped + bmi:age + npreg:bmi:age +
##     bp:bmi:ped
##               Df Deviance AIC  LRT Pr(>Chi)
## <none>               437 469
## glu:ped        1      447 477 9.04   0.0026 **
## npreg:bmi:age  1      447 477 9.35   0.0022 **
## bp:bmi:ped     1      445 475 7.23   0.0072 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Podemos apreciar que entre todos los modelos, el que tiene un valor de AIC/BIC más bajo es el séptimo (interacciones de hasta tercer orden / BIC), sin embargo es el sexto modelo (interaciones de hasta tercer orden / AIC) el que arroja las tasas de error más bajas, sin embargo contiene demasiadas variables que no permite que se haga una interpretación correcta de lo que quiere decir cada una de ellas. Por otro lado observamos que el quinto modelo (interacciones hasta segundo orden / BIC) es bastante simple en cuanto a las relaciones entre variables, además es el segundo modelo que tiene el valor AIC/BIC más bajo. Apelando a la simplicidad de los modelos optaremos por este último.

d) SVM

Analogamente a cuando seleccionamos un modelo para regresión logística, en SVM también existe una forma de considerar el mejor modelo, y es con la función `tune`. Empecemos declarando tres modelos con los tres distintos de kernel (lineal, polinomial y radial) y usemos la función `tune` para elegir los mejores parámetros[7]:

```
set.seed(07042020)
svm_Pima_lin <- tune(svm, type~., data=Pima , kernel ="linear",
                     ranges = list(cost=seq(0.3, 0.4, 0.005)))
svm_Pima_pol <- tune(svm, type~., data=Pima, kernel="polynomial",
                     ranges=list(cost=seq(0.25,0.35, 0.001), degree=c(2:4)))
svm_Pima_rad <- tune(svm, type~., data=Pima,
                     kernel="radial",
                     ranges=list(cost=seq(0.5, 0.7, 0.005),
                                 gamma=seq(0.05, 0.3, 0.05)))
```

Se tiene que los mejores parámetros para el modelo lineal es:

```
##   cost
## 1  0.3
```

Para el modelo polinomial tenemos:

```
##      cost degree
## 176 0.324      3
```

[7]Los rangos usados fueron encontrados probando a partir de rangos más grandes e ir acotando

Finalmente para el radial tenemos que los mejores parámetros son:

```
##     cost gamma
## 14 0.565  0.05
```

Una vez teniendo nuestros mejores modelos calculemos las tasas de error. Se tienen las siguientes tasas de error para el mejor modelo lineal:

```
##      Predicho
## Real   No Yes
##   No  318  37
##   Yes  75 102
```

```
##            No    Yes
## 0.2105 0.1042 0.4237
```

Para el modelo polinomial se tiene:

```
##      Predicho
## Real   No Yes
##   No  347   8
##   Yes 100  77
```

```
##              No     Yes
## 0.20301 0.02254 0.56497
```

Finalmente las tasas para el modelo radial son:

```
##      Predicho
## Real   No Yes
##   No  320  35
##   Yes  73 104
```

```
##              No     Yes
## 0.20301 0.09859 0.41243
```

Podemos ver que el modelo radial es el que tiene las tasas más bajas, seguido del modelo lineal y el modelo polinomial. Por lo tanto nos quedaremos con el modelo radial.

Antes de empezar a calcular las tasas no aparentes hagamos un resumen de los mejores modelos encontrados:

a) LDA: El modelo con interacciones de hasta tercer grado.
b) *Naive Bayes*: El modelo sin las variables `bp, skin` y `age`
c) Regresión Logística: El modelo con interacciones de hasta segundo orden (BIC)
d) SVM: El modelo radial

Ahora, para calcular las tasas no aparentes debemos partir nuestra base en dos conjuntos: *train* y *test*.

Ya teniendo esos conjuntos se entrena el modelo seleccionado con el conjunto *train* y las tasas se calculan con el conjunto *test* con el modelo ya entrenado. Esto se repite muchas veces y el promedio de las tasas obtenidas serán las tasas no aparentes.

Para eso construyamos nuestros conjuntos *train* y *test*. En este caso **dividiremos la base en 0.75 y 0.25 para cada conjunto**, respectivamente.

Observamos que la base tiene 532 observaciones por lo que el conjunto *train* tendra 399 observaciones y el conjunto *test* tendrá las 133 restantes. Este ciclo se repitió 300 veces arrojando las tasas no aparentes siguientes:

Para nuestro modelo obtenido usando análisis de discriminante lineal se tienen las tazas. Se muestra primero la tasa tras una iteración para mostrar como disminuyen conforme el número de iteraciones aumenta:

```r
set.seed(07042020)
(tasas_nap(1, "lda"))
```

```
## [1] 0.2632 0.1596 0.5128
```

```r
lda_aux <-tasas_nap(300, "lda")
(lda_aux)
```

```
## [1] 0.0005764 0.0002622 0.0012121
```

Para nuestro modelo escogido usando *Naive Bayes* se tiene:

```r
set.seed(07042020)
(tasas_nap(1, "naiveBayes"))
```

```
## [1] 0.2180 0.1489 0.3846
```

```r
nB_aux <- tasas_nap(300, "naiveBayes")
(nB_aux)
```

```
## [1] 0.0004511 0.0002247 0.0009091
```

Para el modelo ajustado con regresión logística tenemos:

```r
set.seed(07042020)
(tasas_nap(1, "glm"))
```

```
## [1] 0.2256 0.1596 0.3846
```

```r
glm_aux <- tasas_nap(300, "glm")
(glm_aux)
```

```
## [1] 0.0004010 0.0002622 0.0006818
```

Finalmente usando el kernel radial en SVM tenemos las siguientes tasas:

```r
set.seed(07042020)
(tasas_nap(1, "svm"))
```

```
## [1] 0.2331 0.1277 0.4872
```

```r
svm_aux <- tasas_nap(300, "svm")
(svm_aux)
```

```
## [1] 0.0005013 0.0001124 0.0012879
```

Podemos resumir todas las tazas obtenidas para los mejores modelos en la tabla siguiente:

```
##                                  Tasa de Error Global Tasa de Error NO Tasa de Error YES
## LDA_Aparente                                   0.1861           0.1099            0.3390
## LDA_No_Aparente                                0.0006           0.0003            0.0012
## Naive_Bayes_Aparente                           0.2105           0.1183            0.3955
## Naive_Bayes_No_Aparente                        0.0005           0.0002            0.0009
## Regresión_Logística_Aparente                   0.2105           0.1155            0.4011
## Regresión_Logística_No_Aparente                0.0004           0.0003            0.0007
## SVM_Aparente                                   0.2030           0.0986            0.4124
## SVM_No_Aparente                                0.0005           0.0001            0.0013
```

Tarea 3

II. Considere la base de datos *cad1* del paquete `gRbase`.

```
install.packages("BiocManager")
BiocManager::install("Rgraphviz")
BiocManager::install("graph")
BiocManager::install("RBGL")
BiocManager::install("gRbase")
%install.packages("bnlearn")
library(gRbase)
> str(cad1)
'data.frame': 236 obs. of  14 variables:
 $ Sex        : Factor w/ 2 levels "Female","Male": 2 2 1 2 2 2 2 2 1 2 ...
 $ AngPec     : Factor w/ 3 levels "Atypical","None",..: 2 1 2 2 2 2 2 2 2 1 ...
 $ AMI        : Factor w/ 2 levels "Definite","NotCertain": 2 2 1 2 2 2 2 2 2 2 ...
 $ QWave      : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 2 2 1 1 ...
 $ QWavecode  : Factor w/ 2 levels "Nonusable","Usable": 2 2 2 2 2 2 2 2 1 2 ...
 $ STcode     : Factor w/ 2 levels "Nonusable","Usable": 2 2 2 1 1 1 1 1 1 2 ...
 $ STchange   : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 2 ...
 $ SuffHeartF : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
 $ Hypertrophi: Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
 $ Hyperchol  : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
 $ Smoker     : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
 $ Inherit    : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
 $ Heartfail  : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
 $ CAD        : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
```

Este problema involucra predictores de tipo categórico, no son de tipo numérico. Es un problema de clasificacióon binaria. La variable `CAD` identifica a las dos clases. Dentro del paquete que contiene la base de datos puede leerse informacióon adicional. Considere el ajuste de los modelos o método apropiado:

a) *Naive Bayes*

b) *Regresión logistica*

c) *SVM*

Para cada uno delos tres casos, considere ajustar un modelo o método optimizando su capacidad predictiva, de forma similar al problema anterior.

Reporte las especificaciones de cada modelo o método seleccionado, uno para cada inciso. Reporte de forma tabular las tasas aparentes de clasificación erronea por grupo y la global. También las no aparentes calculadas por el método de repeticiones *training/test*.

Comente brevemente sobre los valores obtenidos por cada modelo seleccionado.

Comenzaremos ajustando cada uno de los modelos, con todas las observaciones.

DATOS

```
dat<-cad1
attach(dat)
```

Donde nuestra variable predictora `CAD` sere nuestra variable predictora, con las siguientes consideraciones.

- Dado que los valores que toma la variable `CAD` es *No* y *Yes* respectivamente, estaremos clasificando las poblaciónes a acorde a ese orden , es decir nuestra primera población serán aquellas observaciones que queden dentro del grupo *No* mientras que nuestra segunda población serán aquellos que contengan el valor *Yes*.

PARTICIÓN DE LOS DATOS

Se dividió nuestros datos para nuestro conjunto de entrenamiento considerando el 75% de los datos totales , el restante serán para las observaciones de prueba.

Dado que nuestro conjunto de datos se encontraba enumerada escogimos una muestra aleatoria que dividie los datos en base a ese elemento de la tabla.

```
set.seed(1)
train<-sample(1:236,177)
datos_train<-dat[train,]
test<-dat[-train,]
```

IMPLEMENTACIÓN DEL MODELO

Ajustando el primer modelo *Naive Bayes* se tiene lo siguiente.

```
ajuste_nb
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##        No       Yes
## 0.5466102 0.4533898
##
## Conditional probabilities:
##      Sex
## Y        Female      Male
##   No  0.2635659 0.7364341
##   Yes 0.1214953 0.8785047
##
##      AngPec
## Y       Atypical       None    Typical
##   No  0.18604651 0.55813953 0.25581395
##   Yes 0.05607477 0.12149533 0.82242991
```

```
##
##      AMI
## Y       Definite NotCertain
##   No  0.09302326 0.90697674
##   Yes 0.47663551 0.52336449
##
##      QWave
## Y            No       Yes
##   No  0.8372093 0.1627907
##   Yes 0.4205607 0.5794393
##
##      QWavecode
## Y      Nonusable     Usable
##   No  0.06976744 0.93023256
##   Yes 0.03738318 0.96261682
##
##      STcode
## Y     Nonusable    Usable
##   No  0.4186047 0.5813953
##   Yes 0.2336449 0.7663551
##
##      STchange
## Y            No       Yes
##   No  0.7131783 0.2868217
##   Yes 0.3831776 0.6168224
##
##      SuffHeartF
## Y            No       Yes
##   No  0.7441860 0.2558140
##   Yes 0.6635514 0.3364486
##
##      Hypertrophi
## Y            No       Yes
##   No  0.6201550 0.3798450
##   Yes 0.8598131 0.1401869
##
##      Hyperchol
## Y            No       Yes
##   No  0.6279070 0.3720930
##   Yes 0.2523364 0.7476636
##
##      Smoker
## Y            No       Yes
##   No  0.3100775 0.6899225
##   Yes 0.1028037 0.8971963
##
##      Inherit
## Y            No       Yes
##   No  0.8062016 0.1937984
##   Yes 0.5420561 0.4579439
##
##      Heartfail
## Y            No       Yes
##   No  0.6589147 0.3410853
```

```
##   Yes 0.8598131 0.1401869
```

Mostrando las probabilidades a priori, es decir $\pi_1 = 0.5466102$ y $\pi_2 = 0.4533898$ mientras que los valores restantes representan la verosimilitud que existe entre cada una de las varibles.

Precisión del modelo ajustado

Con los siguientes datos varemos que tan precisa es la clasificación del modelo considerando todas nuestras observaciones.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  114  17
##        Yes  15  90
##
##                Accuracy : 0.8644
##                  95% CI : (0.814, 0.9054)
##     No Information Rate : 0.5466
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.726
##
##  Mcnemar's Test P-Value : 0.8597
##
##             Sensitivity : 0.8837
##             Specificity : 0.8411
##          Pos Pred Value : 0.8702
##          Neg Pred Value : 0.8571
##              Prevalence : 0.5466
##          Detection Rate : 0.4831
##    Detection Prevalence : 0.5551
##       Balanced Accuracy : 0.8624
##
##        'Positive' Class : No
##
```

Teniendo un p-valor considerablemente bajo podemos decir que nuestro modelo es bastante aceptable por lo que procederemos a ajustar nuestro modelo con nuestras variables de entrenamiento.

Ajuste del modelo con las observaciones de entrenamiento

```
modelo_nb<-naiveBayes(CAD~.,data = dat, subset = train)
modelo_nb
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
```

```
##        No       Yes
## 0.5310734 0.4689266
##
## Conditional probabilities:
##      Sex
## Y        Female      Male
##   No  0.2234043 0.7765957
##   Yes 0.1204819 0.8795181
##
##      AngPec
## Y       Atypical       None    Typical
##   No  0.19148936 0.55319149 0.25531915
##   Yes 0.04819277 0.13253012 0.81927711
##
##      AMI
## Y       Definite NotCertain
##   No  0.07446809 0.92553191
##   Yes 0.51807229 0.48192771
##
##      QWave
## Y            No       Yes
##   No  0.8617021 0.1382979
##   Yes 0.4698795 0.5301205
##
##      QWavecode
## Y      Nonusable     Usable
##   No  0.08510638 0.91489362
##   Yes 0.03614458 0.96385542
##
##      STcode
## Y     Nonusable    Usable
##   No  0.4148936 0.5851064
##   Yes 0.2168675 0.7831325
##
##      STchange
## Y            No       Yes
##   No  0.6914894 0.3085106
##   Yes 0.3855422 0.6144578
##
##      SuffHeartF
## Y           No      Yes
##   No  0.787234 0.212766
##   Yes 0.626506 0.373494
##
##      Hypertrophi
## Y            No       Yes
##   No  0.5638298 0.4361702
##   Yes 0.8915663 0.1084337
##
##      Hyperchol
## Y            No       Yes
##   No  0.6808511 0.3191489
##   Yes 0.2771084 0.7228916
##
```

```
##      Smoker
## Y            No       Yes
##   No  0.3191489 0.6808511
##   Yes 0.1084337 0.8915663
##
##      Inherit
## Y            No       Yes
##   No  0.7978723 0.2021277
##   Yes 0.5301205 0.4698795
##
##      Heartfail
## Y             No        Yes
##   No  0.62765957 0.37234043
##   Yes 0.91566265 0.08433735
```

Podemos notar que las probabilidades a *prori* no difieren mucho con respecto a las del modelo anterior.

Comprobación del modelo

Ahora haremos la comprobación del modelo con respecto a nuestras observaciones de prueba.

```
pred_nb<-predict(modelo_nb,test)
```

Mostrando la matriz de confusión.

```
##
## pred_nb No Yes
##     No  26   6
##     Yes  9  18
```

Precisión del modelo ajustado

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction No Yes
##        No  26   6
##        Yes  9  18
##
##                Accuracy : 0.7458
##                  95% CI : (0.6156, 0.8502)
##     No Information Rate : 0.5932
##     P-Value [Acc > NIR] : 0.01063
##
##                   Kappa : 0.4834
##
##  Mcnemar's Test P-Value : 0.60558
##
##             Sensitivity : 0.7429
##             Specificity : 0.7500
##          Pos Pred Value : 0.8125
##          Neg Pred Value : 0.6667
##              Prevalence : 0.5932
##          Detection Rate : 0.4407
```

```
##    Detection Prevalence : 0.5424
##       Balanced Accuracy : 0.7464
##
##        'Positive' Class : No
##
```

A pesar de únicamente se cuenta con el 25% de todas las observaciones nuestro modelo siguie manteniendo un nivel sumamente alto de precisión ,esto nos conduce que se puede asegurar que las tasas de error serán relativamente bajas.

TASAS DE ERROR

Tasas aparentes y no aparentes de error

Para nuestras observaciones generales la tasas apartentes global y grupal de error están dadas por.

```
tasa_aparente_nb
```

```
##         Tasas aparentes de error
## Global                 0.1355932
## Grupo 0                0.1162791
## Grupo 1                0.1588785
```

Mientras que para nuestras observaciones de entrenamiento y prueba las tasas no aparentes son :

```
tasa_no_aparente_nb
```

```
##         Tasas no aparentes de error
## Global                    0.2542373
## Grupo 0                   0.2571429
## Grupo 1                   0.2500000
```

Podemos observar que efectivamente nuestro modelo contiene unas tasas de error bajas por lo que el nivel de precisión en la clasificación de este modelo es muy exacto.

Regresión Logística

Implementación del modelo

```
modelo_rl<-glm(CAD~.,data = dat, family = binomial(link = "logit"))
summary(modelo_rl)
```

```
##
## Call:
## glm(formula = CAD ~ ., family = binomial(link = "logit"), data = dat)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.2211  -0.4280  -0.1284   0.4520   2.5312
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -1.9907     1.3717  -1.451 0.146690
## SexMale           0.4269     0.5242   0.814 0.415394
## AngPecNone       -0.9167     0.7861  -1.166 0.243542
## AngPecTypical     2.0258     0.6704   3.022 0.002514 **
## AMINotCertain    -1.8638     0.5475  -3.404 0.000663 ***
## QWaveYes          1.7011     0.5516   3.084 0.002043 **
## QWavecodeUsable   0.8020     1.0821   0.741 0.458595
## STcodeUsable     -1.9136     0.7652  -2.501 0.012394 *
## STchangeYes       2.4963     0.6071   4.112 3.92e-05 ***
## SuffHeartFYes     0.3750     0.5581   0.672 0.501609
## HypertrophiYes   -0.7961     0.5992  -1.329 0.183919
## HypercholYes      1.2769     0.4521   2.825 0.004733 **
## SmokerYes         0.3492     0.5504   0.634 0.525844
## InheritYes        0.5909     0.4846   1.219 0.222732
## HeartfailYes     -0.8047     0.6662  -1.208 0.227045
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 325.11  on 235  degrees of freedom
## Residual deviance: 147.73  on 221  degrees of freedom
## AIC: 177.73
##
## Number of Fisher Scoring iterations: 6
```

Podemos notar que las variables individuales al menos en su mayoria no tienen explicación de forma siginificativa al modelo debido a que altos valores en sus p-valores.Sin embargo podemos ver que el valor promedio de la devianza es demasiado bajo por lo que siginica que nuestro modelo en general esta bien ajustado.

Reafirmaremos esto con los niveles de exactitud que tiene

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  116  15
##        Yes  13  92
```

```
##
##                Accuracy : 0.8814
##                  95% CI : (0.8331, 0.9197)
##     No Information Rate : 0.5466
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.7602
##
##  Mcnemar's Test P-Value : 0.8501
##
##             Sensitivity : 0.8992
##             Specificity : 0.8598
##          Pos Pred Value : 0.8855
##          Neg Pred Value : 0.8762
##              Prevalence : 0.5466
##          Detection Rate : 0.4915
##    Detection Prevalence : 0.5551
##       Balanced Accuracy : 0.8795
##
##        'Positive' Class : No
##
```

Nuevamente podemos ver que sus valores de exactitud son bastante altos por lo que el modelo es aceptable .

Ahora pasaremos al ajuste del modelo con las variables de entrenamiento

```
logit<-glm(CAD~.,data = datos_train, family = binomial)

summary(logit)
```

```
##
## Call:
## glm(formula = CAD ~ ., family = binomial, data = datos_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.9992  -0.3334  -0.0593   0.3064   3.4608
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -1.16596    1.85649  -0.628 0.529974
## SexMale         -0.01312    0.70474  -0.019 0.985147
## AngPecNone      -0.34160    1.15036  -0.297 0.766504
## AngPecTypical    3.68073    1.06029   3.471 0.000518 ***
## AMINotCertain   -2.68123    0.75720  -3.541 0.000399 ***
## QWaveYes         1.86956    0.77350   2.417 0.015649 *
## QWavecodeUsable  1.17369    1.39654   0.840 0.400669
## STcodeUsable    -2.91245    1.07188  -2.717 0.006585 **
## STchangeYes      2.83969    0.82276   3.451 0.000558 ***
## SuffHeartFYes    0.41827    0.74885   0.559 0.576465
## HypertrophiYes  -1.20093    0.78798  -1.524 0.127494
## HypercholYes     1.22457    0.61949   1.977 0.048071 *
## SmokerYes        0.16609    0.67747   0.245 0.806332
## InheritYes       0.46319    0.61046   0.759 0.448007
## HeartfailYes    -2.28011    0.98139  -2.323 0.020160 *
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 244.690  on 176  degrees of freedom
## Residual deviance:  88.964  on 162  degrees of freedom
## AIC: 118.96
##
## Number of Fisher Scoring iterations: 7
```

Podemos observar que a diferencia del modelo donde se tomaban en cuenta todas las variables en este modelo hay una variable más que es estadisticamente significativas, además el AIC es menor que el modelo anterior , por lo tanto podemos concluir que este modelo se ajusta mejor que el anterior.

En la matriz de confusión podemos observar lo siguiente.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction No Yes
##        No  30   4
##        Yes  5  20
##
##                Accuracy : 0.8475
##                  95% CI : (0.7301, 0.9278)
##     No Information Rate : 0.5932
##     P-Value [Acc > NIR] : 2.331e-05
##
##                   Kappa : 0.686
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.8571
##             Specificity : 0.8333
##          Pos Pred Value : 0.8824
##          Neg Pred Value : 0.8000
##              Prevalence : 0.5932
##          Detection Rate : 0.5085
##    Detection Prevalence : 0.5763
##       Balanced Accuracy : 0.8452
##
##        'Positive' Class : No
##
```

Tenemos un mejor nivel de exactitud que el modelo anterior, ahora pasaremos a examninar las tasas de error.

Tasas aparentes y no aparentes

Tasas aparentes

```
tasa_aparente_rl
```

```
##         Tasas aparentes de error
## Global                 0.1186441
## Grupo 0                0.1007752
## Grupo 1                0.1401869
```

Y en las no aparentes tenemos.

```
tasa_no_aparente_rl
```

```
##         Tasas no aparentes de error
## Global                    0.1525424
## Grupo 0                   0.1428571
## Grupo 1                   0.1666667
```

En este caso nuestra tasa no aparente de error es un poco más alta que las tasas aparetes sin embargo nuestro nivel de predicción siguie siendo demasiado alto teniendo una efectividad de clasificación de poco más del 84%

Maquinas de Vector Soporte SVM

Ajustando nuestro modelos con todas nuestras observaciones tenemos lo siguiente

```
modelo<-svm(CAD~.,data = dat)

summary(modelo)
```

```
##
## Call:
## svm(formula = CAD ~ ., data = dat)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  118
##
##  ( 59 59 )
##
##
## Number of Classes:  2
##
## Levels:
##  No Yes
```

Podemos notar que por defecto nuestro metodo usa un núcleo radial , si llevamos el ajuste donde nuestras obsercaciones tengan un comportamiento lineal se tiene

```
modelo_lineal<-svm(CAD~.,dat,kernel = "linear",cost = 10, scale = TRUE)

summary(modelo_lineal)
```

```
##
## Call:
## svm(formula = CAD ~ ., data = dat, kernel = "linear", cost = 10,
##     scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
```

```
##        cost:  10
##
## Number of Support Vectors:  79
##
##  ( 41 38 )
##
##
## Number of Classes:  2
##
## Levels:
##  No Yes
```

En este caso nuestro costo fué de 10 podemos notar que hubo un cambio en el número de soporte de los vectores.

Con su matriz de confusion.

```
##       CAD
## predic  No Yes
##    No  122  15
##    Yes   7  92
```

Veamos si el modelo puede mejorar con el metodo de validaciones cruzadas si se incluyen las variables de entrenamiento y de prueba.

```
lineal<-tune(svm,CAD~.,data = dat, kernel ="linear",
             ranges = list( cost=c(0.001 , 0.01 , 0.1, 1 ,5 ,10 ,100)),scale=TRUE)
summary(lineal)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##     1
##
## - best performance: 0.1096014
##
## - Detailed performance results:
##    cost     error dispersion
## 1 1e-03 0.4534420 0.04004721
## 2 1e-02 0.1739130 0.05947378
## 3 1e-01 0.1606884 0.06699933
## 4 1e+00 0.1096014 0.06260566
## 5 5e+00 0.1096014 0.05255532
## 6 1e+01 0.1096014 0.05255532
## 7 1e+02 0.1096014 0.05255532
```

Podemos notar que el costo que tiene el valor 10 contiene un número de error mucho menor que todos los demás costos , por lo tanto seŕa tomado en cuenta para nuestro mejor modelo.

```
m_modelo<-lineal$best.model
m_modelo
```

```
##
## Call:
```

```
## best.tune(method = svm, train.x = CAD ~ ., data = dat, ranges = list(cost = c(0.001,
##     0.01, 0.1, 1, 5, 10, 100)), kernel = "linear", scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##
## Number of Support Vectors:  89
```

Podemos ver los resultados de este modelo en la siguiente matriz de confusion.

```
confusionMatrix(pred_t_o,CAD)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  120  13
##        Yes   9  94
##
##                Accuracy : 0.9068
##                  95% CI : (0.8623, 0.9407)
##     No Information Rate : 0.5466
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8113
##
##  Mcnemar's Test P-Value : 0.5224
##
##             Sensitivity : 0.9302
##             Specificity : 0.8785
##          Pos Pred Value : 0.9023
##          Neg Pred Value : 0.9126
##              Prevalence : 0.5466
##          Detection Rate : 0.5085
##    Detection Prevalence : 0.5636
##       Balanced Accuracy : 0.9044
##
##        'Positive' Class : No
##
```

Se observa que en los resultados que el modelo se adecua perfecfectamente considerando todas las variables, si ajustamos con las variables de entrenamiento y prueba obtenemos lo siguiente.

```
svm_lineal<-tune(svm ,CAD~., data=datos_train , kernel ="linear",
                 ranges = list( cost=c(0.001 , 0.01 , 0.1, 1 ,5 ,10 ,100)),scale=TRUE)

summary(svm_lineal)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
```

```
## - best parameters:
##  cost
##     1
##
## - best performance: 0.09542484
##
## - Detailed performance results:
##    cost      error dispersion
## 1 1e-03 0.46830065 0.09176291
## 2 1e-02 0.16862745 0.06759433
## 3 1e-01 0.11764706 0.07114883
## 4 1e+00 0.09542484 0.05885983
## 5 5e+00 0.10130719 0.06918712
## 6 1e+01 0.10130719 0.06918712
## 7 1e+02 0.10686275 0.06731021
```

Se observa que el costo con el valor de 5 es el que menor error tiene, por tanto será el costo para el mejor modelo.

```
mejor_modelo
```

```
##
## Call:
## best.tune(method = svm, train.x = CAD ~ ., data = datos_train, ranges = list(cost = c(0.001,
##     0.01, 0.1, 1, 5, 10, 100)), kernel = "linear", scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##
## Number of Support Vectors:  65
```

En la matriz de confusiones se tiene.

```
confusionMatrix(predicciones,test$CAD)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction No Yes
##        No  31   6
##        Yes  4  18
##
##                Accuracy : 0.8305
##                  95% CI : (0.7103, 0.9156)
##     No Information Rate : 0.5932
##     P-Value [Acc > NIR] : 8.347e-05
##
##                   Kappa : 0.6441
##
##  Mcnemar's Test P-Value : 0.7518
##
##             Sensitivity : 0.8857
##             Specificity : 0.7500
```

```
##          Pos Pred Value : 0.8378
##          Neg Pred Value : 0.8182
##              Prevalence : 0.5932
##          Detection Rate : 0.5254
##    Detection Prevalence : 0.6271
##       Balanced Accuracy : 0.8179
##
##        'Positive' Class : No
##
```

Nuestra exactidud del modelo deismminuye a comparación del modelo anterior, sin embargo el p-valor siguie siendo demasiado bajo por lo que se toma en cuenta.

Kernel polinomial

Ajustando el kernel del modelo en todas nuestras observaciones a un polinomio se tienen los siguientes resultados.

```
polinomial<-tune("svm" ,CAD~., data=dat , kernel ="polynomial",
                 ranges = list( cost=c(0.001 , 0.01 , 0.1, 1 ,5 ,10 ,100), degree = c(2,3,4)),scale = TR


summary(polinomial)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost degree
##    10      2
##
## - best performance: 0.1103261
##
## - Detailed performance results:
##     cost degree     error dispersion
## 1  1e-03      2 0.4527174 0.07709369
## 2  1e-02      2 0.4527174 0.07709369
## 3  1e-01      2 0.3592391 0.10113626
## 4  1e+00      2 0.1744565 0.08312763
## 5  5e+00      2 0.1231884 0.07170496
## 6  1e+01      2 0.1103261 0.06497904
## 7  1e+02      2 0.1226449 0.07267506
## 8  1e-03      3 0.4527174 0.07709369
## 9  1e-02      3 0.4527174 0.07709369
## 10 1e-01      3 0.4527174 0.07709369
## 11 1e+00      3 0.1909420 0.10866613
## 12 5e+00      3 0.1615942 0.08326679
## 13 1e+01      3 0.1317029 0.06888321
## 14 1e+02      3 0.1436594 0.06601239
## 15 1e-03      4 0.4527174 0.07709369
## 16 1e-02      4 0.4527174 0.07709369
## 17 1e-01      4 0.4527174 0.07709369
## 18 1e+00      4 0.2371377 0.08793419
```

```
## 19 5e+00      4 0.1952899 0.08652267
## 20 1e+01      4 0.1615942 0.07849684
## 21 1e+02      4 0.1480072 0.08010283
```

Para un polinomio de grado 2 con un costo de 10 vemos que este modelo es que el menos error tiene por lo tanto se considera para nuestro mejor modelo.

```
m_modelo_2
```

```
##
## Call:
## best.tune(method = "svm", train.x = CAD ~ ., data = dat, ranges = list(cost = c(0.001,
##     0.01, 0.1, 1, 5, 10, 100), degree = c(2, 3, 4)), kernel = "polynomial",
##     scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  10
##      degree:  2
##      coef.0:  0
##
## Number of Support Vectors:  96
```

```
confusionMatrix(pred_t_o2,CAD)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  120  13
##        Yes   9  94
##
##                Accuracy : 0.9068
##                  95% CI : (0.8623, 0.9407)
##     No Information Rate : 0.5466
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8113
##
##  Mcnemar's Test P-Value : 0.5224
##
##             Sensitivity : 0.9302
##             Specificity : 0.8785
##          Pos Pred Value : 0.9023
##          Neg Pred Value : 0.9126
##              Prevalence : 0.5466
##          Detection Rate : 0.5085
##    Detection Prevalence : 0.5636
##       Balanced Accuracy : 0.9044
##
##        'Positive' Class : No
##
```

Podemos notar que los resultados en comparación con el modelo lineal se mantienen , ahora si consideramos

la selección de nuestras variables por las variables de entrenamiento tenemos lo siguiente. }

```
svm_polinomial<-tune("svm" ,CAD~., data=datos_train , kernel ="polynomial",
                     ranges = list( cost=c(0.001 , 0.01 , 0.1, 1 ,5 ,10 ,100), degree = c(2,3,4)),scale


summary(svm_polinomial)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost degree
##    10      2
##
## - best performance: 0.1130719
##
## - Detailed performance results:
##     cost degree     error dispersion
## 1  1e-03      2 0.4696078 0.08655916
## 2  1e-02      2 0.4696078 0.08655916
## 3  1e-01      2 0.4418301 0.11839911
## 4  1e+00      2 0.1699346 0.10317003
## 5  5e+00      2 0.1189542 0.06731286
## 6  1e+01      2 0.1130719 0.06422767
## 7  1e+02      2 0.1748366 0.08459291
## 8  1e-03      3 0.4696078 0.08655916
## 9  1e-02      3 0.4696078 0.08655916
## 10 1e-01      3 0.4696078 0.08655916
## 11 1e+00      3 0.1980392 0.11833295
## 12 5e+00      3 0.1473856 0.08999232
## 13 1e+01      3 0.1300654 0.06990036
## 14 1e+02      3 0.1803922 0.09346380
## 15 1e-03      4 0.4696078 0.08655916
## 16 1e-02      4 0.4696078 0.08655916
## 17 1e-01      4 0.4696078 0.08655916
## 18 1e+00      4 0.2486928 0.11848577
## 19 5e+00      4 0.2035948 0.11800006
## 20 1e+01      4 0.1696078 0.08444128
## 21 1e+02      4 0.1529412 0.08528514
```

En virtud de que tiene el menor error el mejor modelo se ajustará a un polinomio de grado 2 con un costo de 5

```
confusionMatrix(predicciones_2,test$CAD)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction No Yes
##        No  27   4
##        Yes  8  20
##
##                Accuracy : 0.7966
```

```
##                  95% CI : (0.6717, 0.8902)
##     No Information Rate : 0.5932
##     P-Value [Acc > NIR] : 0.0007713
##
##                   Kappa : 0.5893
##
##  Mcnemar's Test P-Value : 0.3864762
##
##             Sensitivity : 0.7714
##             Specificity : 0.8333
##          Pos Pred Value : 0.8710
##          Neg Pred Value : 0.7143
##              Prevalence : 0.5932
##          Detection Rate : 0.4576
##    Detection Prevalence : 0.5254
##       Balanced Accuracy : 0.8024
##
##        'Positive' Class : No
##
```

En comparación del modelo lineal notamos que hay hay un pequeño desajuste pues hay una menor cantidad de asignaciones en los valores acertados en el modelo polonomial a comparación del modelo lineal.

Modelo radial

Finalmente en el modelo radial ajustandolo a todas nuestras observaciones se tiene que

```
radial<- tune(svm , CAD~., data = dat, kernel ="radial",
              ranges = list( cost=c (0.1 ,1 ,10 ,100 ,1000) ,
                             gamma =c(0.5 ,1 ,2 ,3 ,4) ),scale = TRUE)

summary(radial)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##     1   0.5
##
## - best performance: 0.1474638
##
## - Detailed performance results:
##     cost gamma     error dispersion
## 1  1e-01   0.5 0.2068841 0.08084935
## 2  1e+00   0.5 0.1474638 0.05518808
## 3  1e+01   0.5 0.1563406 0.06745496
## 4  1e+02   0.5 0.1563406 0.06745496
## 5  1e+03   0.5 0.1563406 0.06745496
## 6  1e-01   1.0 0.3500000 0.12847937
## 7  1e+00   1.0 0.1518116 0.05509947
## 8  1e+01   1.0 0.1606884 0.05093305
## 9  1e+02   1.0 0.1606884 0.05093305
## 10 1e+03   1.0 0.1606884 0.05093305
```

```
## 11 1e-01   2.0 0.4519928 0.10999707
## 12 1e+00   2.0 0.2369565 0.05197308
## 13 1e+01   2.0 0.2112319 0.06503374
## 14 1e+02   2.0 0.2112319 0.06503374
## 15 1e+03   2.0 0.2112319 0.06503374
## 16 1e-01   3.0 0.4519928 0.10999707
## 17 1e+00   3.0 0.2875000 0.07593848
## 18 1e+01   3.0 0.2791667 0.07491550
## 19 1e+02   3.0 0.2791667 0.07491550
## 20 1e+03   3.0 0.2791667 0.07491550
## 21 1e-01   4.0 0.4519928 0.10999707
## 22 1e+00   4.0 0.2960145 0.07116489
## 23 1e+01   4.0 0.2960145 0.07116489
## 24 1e+02   4.0 0.2960145 0.07116489
## 25 1e+03   4.0 0.2960145 0.07116489
```

Donde podemos ver que para $\gamma = 0.5$ y un costo de 1 tenemos el mejor modelo ya que es el que tiene el minimo error

```
confusionMatrix(pred_t_o3,CAD)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  121   4
##        Yes   8 103
##
##                Accuracy : 0.9492
##                  95% CI : (0.9129, 0.9735)
##     No Information Rate : 0.5466
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8977
##
##  Mcnemar's Test P-Value : 0.3865
##
##             Sensitivity : 0.9380
##             Specificity : 0.9626
##          Pos Pred Value : 0.9680
##          Neg Pred Value : 0.9279
##              Prevalence : 0.5466
##          Detection Rate : 0.5127
##    Detection Prevalence : 0.5297
##       Balanced Accuracy : 0.9503
##
##        'Positive' Class : No
##
```

Para el ajuste del modelo radial con el método de validaciones cruzadas su nivel de exactidud es mucho más eficaz a comparación de los metodos anteriores.

Para las variables de entrenamiento se tiene que.

```
svm_radial<- tune(svm , CAD~., data = datos_train, kernel ="radial",
                  ranges = list( cost=c (0.1 ,1 ,10 ,100 ,1000) ,
                                 gamma =c(0.5 ,1 ,2 ,3 ,4) ),scale = TRUE)
```

```r
summary(svm_radial)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##     1   0.5
##
## - best performance: 0.1300654
##
## - Detailed performance results:
##     cost gamma     error dispersion
## 1  1e-01   0.5 0.2104575 0.13201248
## 2  1e+00   0.5 0.1300654 0.10837329
## 3  1e+01   0.5 0.1578431 0.09321271
## 4  1e+02   0.5 0.1578431 0.09321271
## 5  1e+03   0.5 0.1578431 0.09321271
## 6  1e-01   1.0 0.4375817 0.19052565
## 7  1e+00   1.0 0.1581699 0.09730053
## 8  1e+01   1.0 0.1692810 0.11083095
## 9  1e+02   1.0 0.1692810 0.11083095
## 10 1e+03   1.0 0.1692810 0.11083095
## 11 1e-01   2.0 0.4712418 0.16176966
## 12 1e+00   2.0 0.2150327 0.09208241
## 13 1e+01   2.0 0.2039216 0.08943348
## 14 1e+02   2.0 0.2039216 0.08943348
## 15 1e+03   2.0 0.2039216 0.08943348
## 16 1e-01   3.0 0.4712418 0.16176966
## 17 1e+00   3.0 0.2944444 0.09809306
## 18 1e+01   3.0 0.2663399 0.08793556
## 19 1e+02   3.0 0.2663399 0.08793556
## 20 1e+03   3.0 0.2663399 0.08793556
## 21 1e-01   4.0 0.4712418 0.16176966
## 22 1e+00   4.0 0.3058824 0.11212320
## 23 1e+01   4.0 0.3114379 0.11497459
## 24 1e+02   4.0 0.3114379 0.11497459
## 25 1e+03   4.0 0.3114379 0.11497459
```

Para $\gamma = 1$ y un costo de 1, se obtiene el menor error y por lo tanto el mejor modelo.

```r
confusionMatrix(predicciones_3,test$CAD)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction No Yes
##        No  26   5
##        Yes  9  19
##
##                Accuracy : 0.7627
```

```
##                  95% CI : (0.6341, 0.8638)
##     No Information Rate : 0.5932
##     P-Value [Acc > NIR] : 0.004836
##
##                   Kappa : 0.5209
##
##  Mcnemar's Test P-Value : 0.422678
##
##             Sensitivity : 0.7429
##             Specificity : 0.7917
##          Pos Pred Value : 0.8387
##          Neg Pred Value : 0.6786
##              Prevalence : 0.5932
##          Detection Rate : 0.4407
##    Detection Prevalence : 0.5254
##       Balanced Accuracy : 0.7673
##
##        'Positive' Class : No
##
```

En contraste al modelo anterior, aqui el nivel de exactitud es menor a comparación de todos los demás métodos.

TASAS DE ERROR

Para las tasas apartentes tenemos los siguientes datos resumidos.

```
tasa_aparente
```

```
##                Global    Grupo 0    Grupo 1
## Lineal     0.09322034 0.06976744 0.12149533
## Polinomial 0.09322034 0.06976744 0.12149533
## Radial     0.05084746 0.06201550 0.03738318
```

Si consideramos todo el conjunto de nuestras observaciones, los valoes de nuestros errores son relativamente bajos, en consecuencia, para el modelo con el metodo radial es que tiene el menor error global y grupal, y por lo tanto sería la mejor opción entre los 3.

En las tasas no aparentes se obtuvieron los siguientes resultados.

```
tasa_no_aparente
```

```
##               Global   Grupo 0   Grupo 1
## Lineal     0.1694915 0.1142857 0.2500000
## Polinomial 0.2033898 0.2285714 0.1666667
## Radial     0.2372881 0.2571429 0.2083333
```

Muy por el contrario que en el caso de considerar todas nuestras observaciones, en este caso el método radial parece ser la peor opción a comparción de los otros 2 metodos, en este caso el método lineal es la mejor opción.

Comparación de tasas

Para las tasas Aparentes se tiene

Método	Global	Grupo 0	Grupo 1
SVM Lineal	0.09322034	0.06976744	0.12149533
SVM Polonomial	0.09322034	0.06976744	0.12149533
SVM Radial	0.05084746	0.06201550	0.03738318
Regresión Logística	0.1186441	0.1007752	0.1401869
Naive Bayes	0.1355932	0.1162791	0.1588785

Para las tasas no aparentes tenemos

Método	Global	Grupo 0	Grupo 1
SVM Lineal	0.1694915	0.1142857	0.2500000
SVM Polonomial	0.2033898	0.2285714	0.1666667
SVM Radial	0.2372881	0.2571429	0.2083333
Regresión Logística	0.1525424	0.1428571	0.1666667
Naive Bayes	0.2542373	0.2571429	0.2500000

Podemos concluir que para las tasas aparentes el ajustando el modelo de `SVM` con el método radial obtenemos un error mucho más bajo a comparación de los demás modelo , lo que implica que este es el mejor modelo a ajustar.

Mientras que para las tasas no aparentes , la Regresión logística parece ser que tiene en proporción con sus tasas globales y grupal le menor tasas lo que para un cojunto de entremamiento y prueba este es el mejor modelo que se puede ajustar.

Ejercio 3

III. Considere la base de datos *Glucose1.txt* **disponible en classroom. Con modificaciones menores:**

```
## 'data.frame':    145 obs. of  7 variables:
##  $ Patient       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Weight        : num  0.81 0.95 0.94 1.04 1 0.76 0.91 1.1 0.99 0.78 ...
##  $ Fglucose      : int  80 97 105 90 90 86 100 85 97 97 ...
##  $ GlucoseInt    : int  356 289 319 356 323 381 350 301 379 296 ...
##  $ InsulinResp   : int  124 117 143 199 240 157 221 186 142 131 ...
##  $ InsulineResist: int  55 76 105 108 143 165 119 105 98 94 ...
##  $ Class         : int  3 3 3 3 3 3 3 3 3 3 ...
```

1. Este problema involucra cinco predictores de tipo continuo o numérico. Es un problema de clasificación de tres grupos. *Class* **identifica a las tres clases. Considere el ajuste de un modelo de regresión logística. El ajuste lo puede hacer utilizando, e.g., alguno de los dos paquetes listados abajo. Elija uno de los tres grupos como grupo de referencia y justifique su elección.**

a) La función *multinorm*() **del paquete** *nnet*.

b) *vglm*() **del paquete** *VGAM*.

Decidimos utilizar el *Grupo* 3 como grupo de referencia ya que este es el que corresponde a las personas *normales*, es decir que no tienen ningún tipo de diabetes, y tomando en cuenta que, en los estudios realizados por cluster[1], el grupo *normal* separaba a los dos restantes, nos parece lo más adecuado.

Considere optimizar la selección de un modelo desde dos puntos de vista diferentes aunque independientes:

a) *Descriptivo*. **Considerar su interpretación. Para hacer inferencia.**

b) *Predictivo*. **Optimizando su capacidad predictiva.**

Es factible que ofrezca dos modelos diferentes, aunque pueden coincidir. Para el modelo descriptivo, comente sobre su interpretación. No es fácil, son tres clases y el ajuste de dos regresiones binarias no independientes. Para el predictivo, optimice las tasas de error de clasificación. En ambos casos reporte el modelo ajustado y las tasas de clasificación errónea por clase y global. Aparentes y no aparentes.

Primero, como se pude observar arriba, *R* toma la columna *Class* como *int* y nosotros necesitamos que lo tome como *factor*, y como ya vimos que el número 3 es el que corresponde a las personas *normales*, usamos la función *relevel*() para que tome al número 3 como el de referencia y no al 1 (el 1 significa overt diabetes, el 2 chemical diabetes).

```
data$Class<-factor(data$Class)
data$Class<-relevel(data$Class, ref="3")
str(data)
```

```
## 'data.frame':    145 obs. of  7 variables:
##  $ Patient       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Weight        : num  0.81 0.95 0.94 1.04 1 0.76 0.91 1.1 0.99 0.78 ...
##  $ Fglucose      : int  80 97 105 90 90 86 100 85 97 97 ...
```

[1]Estudios que se describen en el *PDF* que se adjuntó para este ejercicio.

```
##  $ GlucoseInt    : int  356 289 319 356 323 381 350 301 379 296 ...
##  $ InsulinResp   : int  124 117 143 199 240 157 221 186 142 131 ...
##  $ InsulineResist: int  55 76 105 108 143 165 119 105 98 94 ...
##  $ Class         : Factor w/ 3 levels "3","1","2": 1 1 1 1 1 1 1 1 1 1 ...
```

a) *Descriptivo*. **Considerar su interpretación. Para hacer inferencia.**

Como para este modelo solo buscamos hacer inferencia, no es necesario modificar o particionar los datos de la tabla, por lo que le aplicaremos una regresión logística multinomial y analizaremos los resultados.

```
library(nnet)
modelo<-multinom(data$Class~Weight+Fglucose+GlucoseInt+InsulinResp+InsulineResist, data=data)
```

```
summary(modelo)
```

```
## Call:
## multinom(formula = data$Class ~ Weight + Fglucose + GlucoseInt +
##     InsulinResp + InsulineResist, data = data)
##
## Coefficients:
##   (Intercept)    Weight   Fglucose GlucoseInt  InsulinResp InsulineResist
## 1   -208.4327 -82.07548  0.9239653  0.3427711 -0.008435929      0.1494886
## 2   -102.0322 -11.64981 -0.3596835  0.3217209 -0.017688962      0.1008461
##
## Std. Errors:
##   (Intercept)      Weight  Fglucose GlucoseInt InsulinResp InsulineResist
## 1   0.1004119  0.01763537 1.3959616 0.23041674  0.05935010     0.13463592
## 2   4.2497570 19.91614152 0.2762284 0.05751652  0.04213159     0.06589764
##
## Residual Deviance: 1.073039
## AIC: 25.07304
```

Para este análisis descriptivo podemos decir varias cosas; Cada renglón en la tabla de coeficientes corresponde a la ecuación del modelo. El primer renglón representa los coeficientes del resultado tipo 1 (diabetes manifiesta) en comparación con nuestra referencia que es el tipo 3. El segundo renglón representa los coeficientes del resultado tipo 2 (diabetes química) en comparación con nuestra referencia (tipo 3).

Notemos que estamos obteniendo los coeficientes con los logitos entonces los transformamos para poder interpretar "mejor"

```
exp(coef(modelo))
```

```
##    (Intercept)       Weight  Fglucose GlucoseInt InsulinResp
## 1 3.011691e-91 2.265011e-36 2.5192603   1.408846   0.9915996
## 2 4.875178e-45 8.720716e-06 0.6978972   1.379500   0.9824666
##   InsulineResist
## 1       1.161240
## 2       1.106106
```

De lo que podremos decir que la razón de momios para un aumento de una unidad en la variable $Weight$ es de 2.26×10^{-36} de ser tipo 1 vs ser tipo 3. De manera análoga se pueden analizar los demás coeficientes. En resumen, un valor mayor a 1 representa un aumento, un valor igual a 1 representa que no hubo cambios y un valor menor a 1 representa que hubo una disminución.

De igual modo, con los coeficientes obtenemos las siguientes ecuaciones:

$$y_1 = ln\left(\frac{P(1)}{P(3)}\right) = -208.43 - 82.07(Weight) + 0.92(Fglucose) + 0.34(GlucoseInt) - 0.01(InsulinResp) + 0.15(InsulineResist)$$

$$y_2 = ln\left(\frac{P(2)}{P(3)}\right) = -102.03 - 11.65(Weight) - 0.36(FGlucose) + 0.32(GlucoseInt) - 0.02(InsulinResp) + 0.10(InsulineResist)$$

De esta forma, podemos decir que

$$\frac{P(1)}{P(3)} = e^{y_1} \quad \text{y} \quad \frac{P(2)}{P(3)} = e^{y_2}$$

Así,

$$\frac{P(1)+P(2)}{P(3)} = e^{y_1} + e^{y_2}; \quad P(1)+P(2)+P(3) = 1$$

Lo que implica que,

$$\frac{1-P(3)}{P(3)} = e^{y_1} + e^{y_2} \Longrightarrow \frac{1}{P(3)} = 1 + e^{y_1} + e^{y_2}$$

Obteniendo

$$P(3) = \frac{1}{1+e^{y_1}+e^{y_2}}$$

Y sustituyendo esto en las ecuaciones anteriores podemos decir que

$$P(1) = \frac{e^{y_1}}{1+e^{y_1}+e^{y_2}}; \qquad P(2) = \frac{e^{y_2}}{1+e^{y_1}+e^{y_2}}$$

y estos valores se obtienen con

```
predict(modelo, data, type = "prob")
```

A su vez, podemos ver la matriz de confusión

```
matconf <- table(predict(modelo,data), data$Class, dnn = c("Predicciones", "Clases verdaderas"))
matconf
```

```
##             Clases verdaderas
## Predicciones  3  1  2
##            3 76  0  0
##            1  0 33  0
##            2  0  0 36
```

∴ Podemos decir que, aparentemente, este es un excelente modelo ya que no hubo error de clasificación en ninguno de los casos.

---

b) *Predictivo*. **Optimizando su capacidad predictiva.**

Una vez que tenemos esto, debemos dividir la muestra en *Train* y *Test*, como tenemos 145 datos, decidimos usar $\frac{2}{3}$ de los datos para train (97 pacientes) y $\frac{1}{3}$ para test (48 pacientes), pero como estos datos deben ser proporcionales, necesitamos saber cuántos tenemos de cada clase, dividirla en 3 y hacer las clasificaciones.

```
sum(data$Class==1) ##=33-overt diabetes
```

```
## [1] 33
```

```
sum(data$Class==2) ##=36-chemical diabetes
```

```
## [1] 36
```

```
sum(data$Class==3) ##=76-normal
```

```
## [1] 76
```

Una vez que sabemos cuántos tenemos de cada uno lo dividiremos de la siguiente forma:

Variable	Train	Test	Total
Normal	51	25	76
Overt diabetes	22	11	33
Chemical diabetes	24	12	36
Total	97	48	145

Y para que sea más fácil dividirlos así, ordenamos la tabla de acuerdo a la variable *Class* y hacemos 2 sub-dataframes

```
data1= data %>% arrange(Class)
data1[c(72:79,109:111),]##obtenemos algunos datos de la tabla, para ver que sí está ordenada por clase.
d.train=data1[c(1:51,77:98,110:133),]##sub-dataframe de los datos para train
d.train
d.test=data1[c(52:76,99:109,134:145),] ##sub-dataframe de los datos para test
d.test
```

Ya que tenemos esto podríamos pensar en mejorar el modelo descriptivo quitando o modificando alguna de las variables. Como no somos expertos en el tema que trata la base y no sabemos a ciencia cierta qué tan importante es una variable con respecto a la otra aplicaremos la función *step*() para tratar de reducir el *AIC*.

```
stm <- step(modelo)
```

```
stm$coefnames
```

```
## [1] "(Intercept)"    "Fglucose"       "GlucoseInt"     "InsulineResist"
```

```
stm$AIC
```

```
## [1] 16.89079
```

∴ Con este procedimiento obtuvimos que las variables más importantes son *Fglucose*, *GlucoseInt* e *InsulinResist*, por lo que serán las que utilizaremos para el modelo predictivo.

```
library(nnet)
modelo2<-multinom(d.train$Class~Fglucose+GlucoseInt+InsulineResist, data=d.train)
```

```
summary(modelo2)
```

```
## Call:
## multinom(formula = d.train$Class ~ Fglucose + GlucoseInt + InsulineResist,
##     data = d.train)
##
## Coefficients:
##   (Intercept)      Fglucose GlucoseInt InsulineResist
## 1  -121.80213  0.4941832164  0.1404941     0.03818006
## 2   -63.94284 -0.0007602191  0.1414042     0.03972744
##
## Std. Errors:
```

```
##   (Intercept)  Fglucose GlucoseInt InsulineResist
## 1 0.003551946 0.3992967 0.09879199     0.05331190
## 2 0.001564397 0.3713650 0.09393903     0.04869824
##
## Residual Deviance: 1.298077
## AIC: 17.29808
```

Aquí podemos ver que efectivamente el AIC baja de 25.07 a 17.3, lo cual indica una mejora en el modelo. Ahora le aplicamos el modelo a los datos $Test$

```
p<-predict(modelo2,d.test)##predecimos la clasificación de _d.test_ con el modelo de train #vector de p
dc<-data1[c(52:76,99:109,134:145),]$Class ##vector de clasificación real de esos datos
sum(dc==p) ##saber cuántos coinciden
```

```
## [1] 44
```

```
cm=table(p,dc, dnn = c("Predicciones", "Clases verdaderas"))
cm
```

```
##             Clases verdaderas
## Predicciones  3  1  2
##            3 21  0  0
##            1  0 11  0
##            2  4  0 12
```

Podemos ver que es un buen modelo, ya que predijo bien 44/48

$\therefore$ Es un buen modelo ya que sólo clasificó mal los del Grupo 3 (Normales) con los del Grupo 2 (Chemical Diabetes), lo cual no debería sorprendernos tanto ya que los pacientes con "Chemical Diabetes" son los que no presentan sintomas de tener diabetes, aunque sí la tengan y esto podría coincidir con los estudios de alguien sin diabetes.

Ahora vamos a sacar los errores $No\ Aparentes$, empezando por la Tasa Global; que es todos los que fueron mal clasificados entre el número total de observaciones

```
##Tasa global
1-sum(diag(cm))/sum(cm)
```

```
## [1] 0.08333333
```

$\therefore$ La Tasa Global de error es de 8.33%

Ahora, para las tasas pro Grupo; son las observaciones mal clasificadas del grupo, entre las observaciones totales del grupo. Como vimos solo hubo mal clasificados en el grupo 3, entonces solo sacaremos esta tasa.

```
#Tasas pro grupo
###Tasa grupo 3
4/25
```

```
## [1] 0.16
```

$\therefore$ La Tasa de error pro Grupo3 es de 16%

Finalmente el modelo quedaría como:

$$ln\left(\frac{P(1)}{P(3)}\right) = -121.8 + 0.49(Fglucose) + 0.14(GlucoseInt) + 0.04(InsulineResist)$$

$$ln\left(\frac{P(2)}{P(3)}\right) = -63.94 - 0.0007(FGlucose) + 0.14(GlucoseInt) + 0.03(InsulineResist)$$

con:

$$Tasas - de - error = \begin{cases} Global = 8.33\% \\ Grupo3 = 16\% \\ Grupo1 = Grupo2 = 0\% \end{cases}$$

Para sacar los errores *Aparentes*, entrenaremos el modelo con todos los datos y lo probaremos, de nuevo, con todos los datos para después sacar las tasas de error.

```
library(nnet)
modelo3<-multinom(data1$Class~Fglucose+GlucoseInt+InsulineResist, data=data1)
```

```
summary(modelo3)
```

```
## Call:
## multinom(formula = data1$Class ~ Fglucose + GlucoseInt + InsulineResist,
##     data = data1)
##
## Coefficients:
##   (Intercept)   Fglucose GlucoseInt InsulineResist
## 1   -295.6374  1.1321101  0.3607800     0.05851646
## 2   -144.6012 -0.3013874  0.3879609     0.06115037
##
## Std. Errors:
##   (Intercept)  Fglucose GlucoseInt InsulineResist
## 1 0.007051746 0.6031367 0.10891053     0.09811106
## 2 0.006912303 0.3644583 0.07344987     0.05097576
##
## Residual Deviance: 0.898712
## AIC: 16.89871
```

Podemos ver que usando todos los datos tenemos una mejora en el modelo ya que el AIC bajó nuevamente, de 17.3 a 16.9.

Ahora vamos a probar el modelo para todos los datos

```
p1<-predict(modelo3,data) #vector de predicción
dc1<-data$Class ##vector de clasificación real de esos datos
sum(dc1==p1) ##saber cuántos coinciden
```

```
## [1] 145
```

```
cm1=table(p1,dc1, dnn = c("Predicciones", "Clases verdaderas"))
cm1
```

```
##             Clases verdaderas
## Predicciones  3  1  2
##            3 76  0  0
##            1  0 33  0
##            2  0  0 36
```

Podemos ver que este es un excelente modelo ya que clasificó correctamente todos los datos.Por lo tanto las tasas de error son cero.

$\therefore$ El modelo quedaría como:

$$ln\left(\frac{P(1)}{P(3)}\right) = -295.6 + 1.13(Fglucose) + 0.36(GlucoseInt) + 0.05(InsulineResist)$$

$$ln\left(\frac{P(2)}{P(3)}\right) = -144.60 - 0.3(FGlucose) + 0.38(GlucoseInt) + 0.06(InsulineResist)$$