Trabajo de Fin de Máster PEC 2: Desarrollo del trabajo. Fase 1.

Pablo Núñez de Arenas Martínez

24/4/2023

Índice:

Tal y como comentamos en la anterior entrega la implementación de la mayoría de los modelos de aprendizaje automático requieren de:

1. Recopilar datos.
2. Explorar y preprocesar datos.
3. Construir y entrenar un modelo en los datos.
4. Evaluar el rendimiento del modelo.
5. Mejorar el rendimiento del modelo.
6. Evaluar el modelo mejorado.

Explorar y preprocesar datos:

Esta sección contiene un breve resumen de las variables y una serie de gráficos que nos permiten ver la relación de algunas variables con la respuesta, después se aplicarán técnicas de reducción de dimensionalidad para reducir el número de variables y evitarnos problemas de multicolinealidad.

Exploración de la base de datos:

Recordemos que de las 29 variables iniciales en la entrega anterior acordamos eliminar las variables “affected”, “bp.limit”, “GRF” y “stage” finalmente se borrará la variable “class” en vez de “affected” ambas variables vienen a representar lo mismo, “class” hace referencia al número de pacientes que padecen o no CKD con dos variables de tipo carácter y la variable “affected” es la misma pero en forma binaria, lo cual nos puede ser más útil a la hora de probar algunos modelos que no admitan entradas de datos de tipo carácter. En este caso cuando affected sea igual a 0 significará que el paciente está sano y cuando affected sea igual a 1 significará que el paciente sufre de CKD.

Resumen del dataset:

```
affected sg     al      rbc     su      pc      pcc     ba          bgr
0: 72    0: 3   0:116   0:175   0:170   0:155   0:173   0:189   1      :79
1:128    1:45   1: 21   1: 25   1:  6   1: 45   1: 27   1: 11   0      :70
         2:36   2: 27           2:  9                           3      :14
         3:75   3: 23           3:  8                           2      :13
         4:41   4: 13           4:  6                           4      :11
                                5:  1                           5      : 4
                                                                (Other): 9
      bu           sod     sc      pot          hemo         pcv
0      :108   4      :92   0:159   0:197   5      :49   6      :56
1      : 53   5      :49   1:  4   1:  1   4      :28   7      :29
2      : 16   6      :22   2:  1   2:  1   7      :26   5      :23
3      : 11   3      :14   3: 22   3:  1   3      :23   4      :22
4      :  5   7      : 9   4:  9           8      :20   9      :19
5      :  5   2      : 6   5:  4           6      :19   3      :18
(Other):  2   (Other): 8   6:  1           (Other):35   (Other):33
```

```
      rbcc          wbcc     htn      dm       cad      appet    pe       ane
4        :96   6       :98   0:122    0:130    0:178    0:160    0:165    0:168
5        :23   5       :47   1: 78    1: 70    1: 22    1: 40    1: 35    1: 32
2        :21   7       :29
3        :21   0       :10
6        :18   1       : 6
7        : 9   2       : 6
(Other):12   (Other): 4
      age
7        :48
8        :34
6        :33
5        :31
3        :14
4        :12
(Other):28
```

Ilustración 1. Resumen de la base de datos.

Primero podemos observar como la variable respuesta esta descompensada, la desigualdad en la distribución de la variable respuesta puede afectar el rendimiento del modelo y la precisión de las predicciones. Solucionaremos este problema mediante el sobremuestreo sintético de la clase mayoritaria.

Por otro lado, podemos ver cómo hay varias variables con varios niveles donde las tablas de contingencia están sumamente desbalanceadas, como por ejemplo los niveles de albumina (al) o los de azúcar (su).

Técnicas de reducción de multidimensionalidad:

Análisis de correspondencia múltiple (MCA):

Primero aplicaremos una técnica de Análisis de Correspondencia Múltiple (MCA), una técnica de reducción de dimensionalidad que se utiliza para analizar la relación entre variables categóricas en un conjunto de datos. En este caso excluiremos la variable respuesta ya que lo que queremos es analizar la relación entre las variables predictoras.

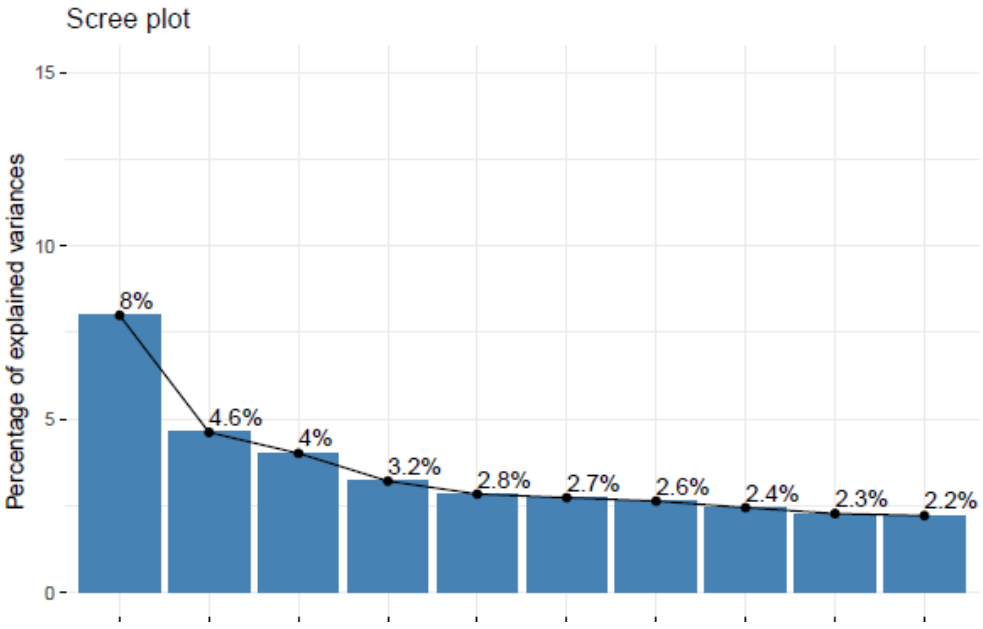


Ilustración 2. Variabilidad explicada por cada dimensión.

Vemos como Las 10 primeras dimensiones apenas abarcan un 35% de la variabilidad siendo necesario abarcar 31 dimensiones para llegar a explicar el 70% de la variabilidad.

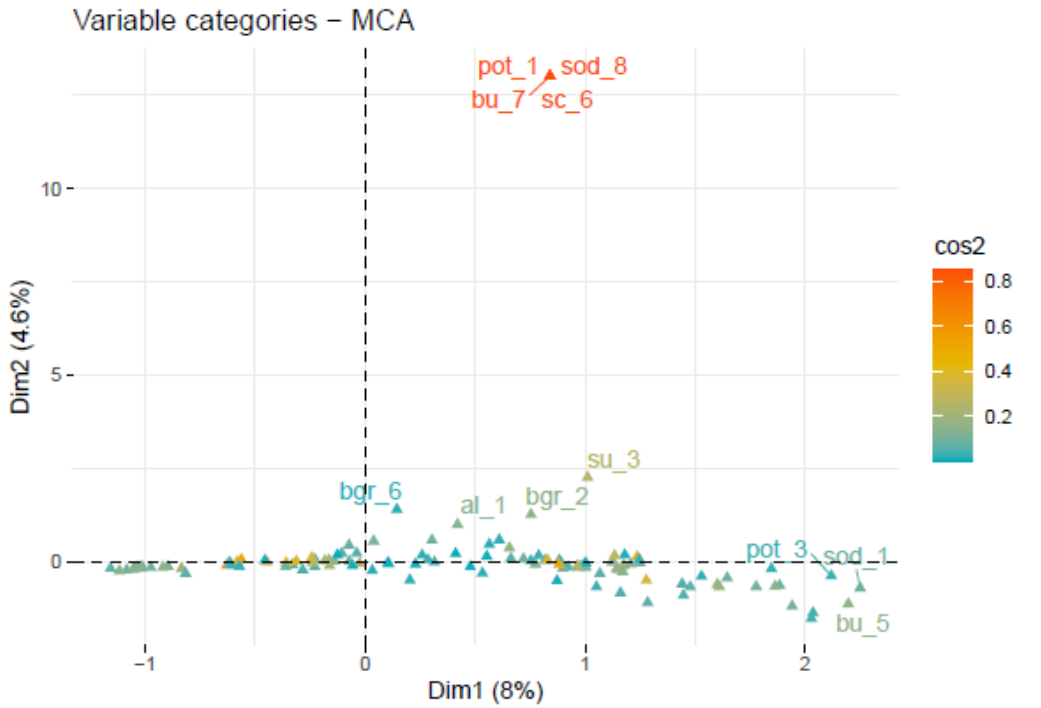


Ilustración 3. Representación de los individuos y variables sobre las dos primeras dimensiones.

Existe mucho solapamiento de variables que no parecen aportar mucha información. He de destacar que esta representación tampoco es una representación muy fiable del peso de las variables ya que ambas dimensiones acumulan menos de un 13% de la variabilidad.

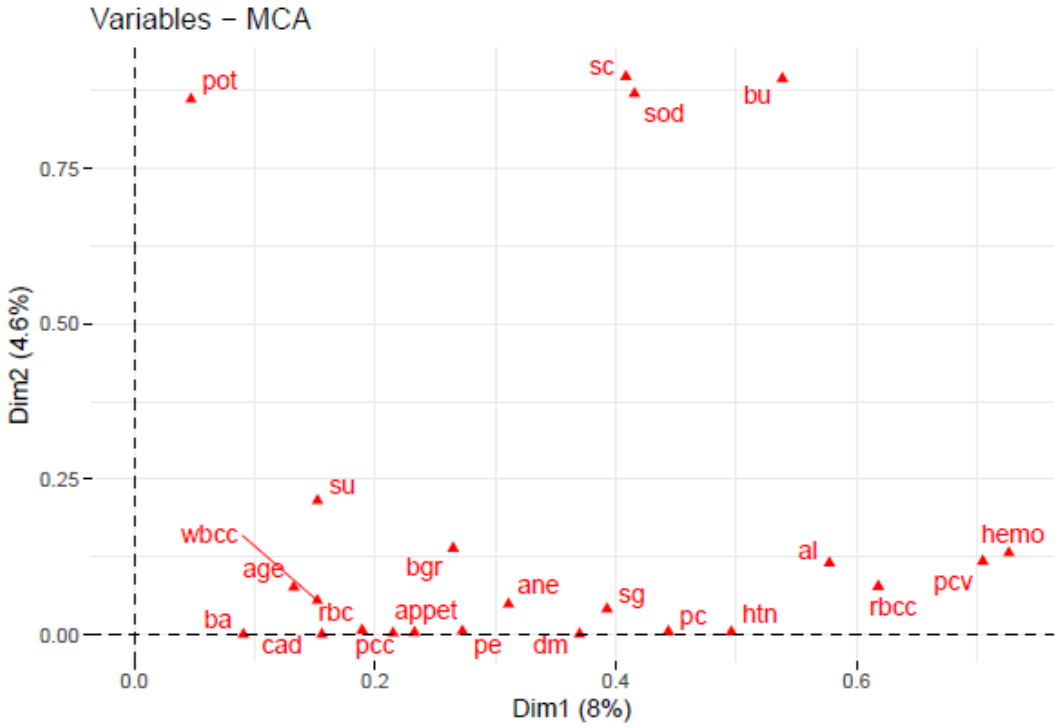


Ilustración 4. Correlación entre las 2 primeras dimensiones y cada variable.

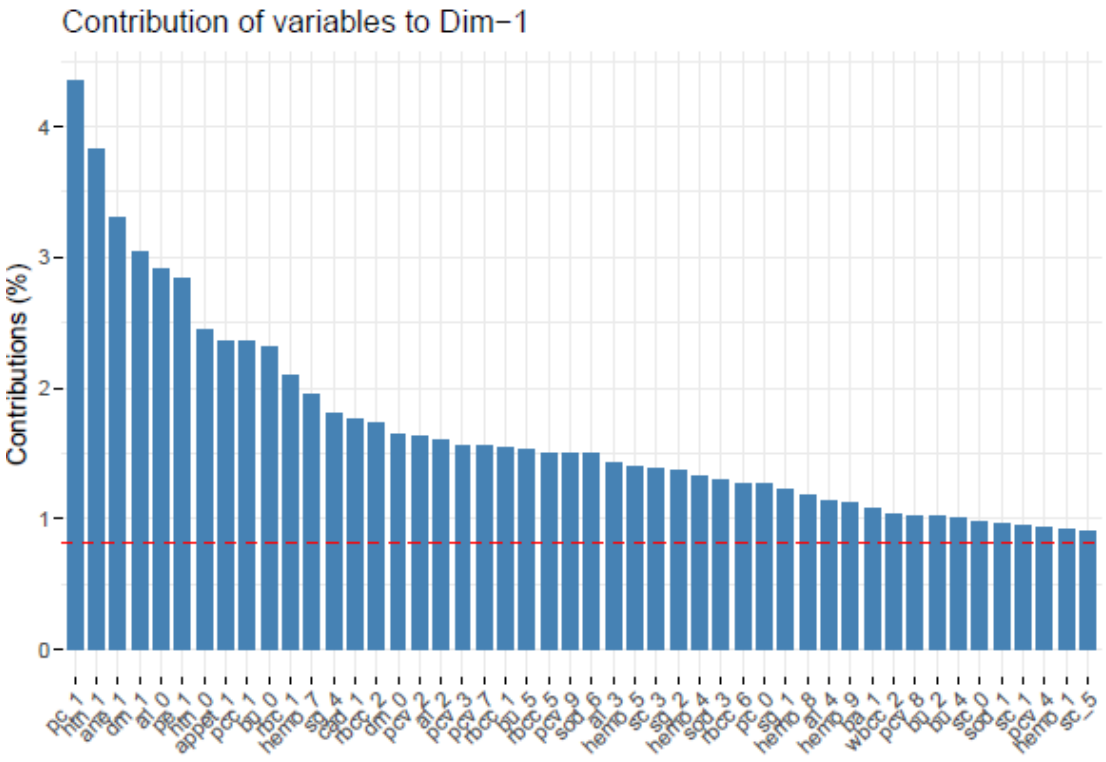
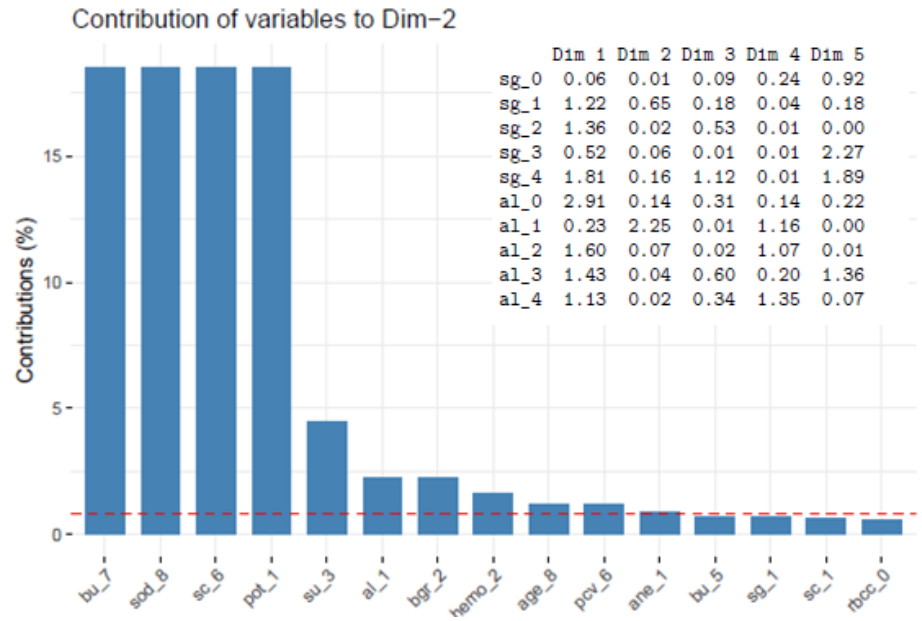


Ilustración 5. Peso de cada nivel de cada variable sobre la primera dimensión.

Sobre la primera dimensión podemos destacar variables como la hipertensión (htn) cuyos dos niveles aparecen en segunda como séptima posición, ambos niveles de pus cells (pc) aparecen en el top 40 al igual que la diabetes melitus (dm), la anemia o la falta de apetito.

La albumina solo aparecen los niveles 0, 2 3 y 4 en el top 40 además de estar muy descompensada. Pedal anemia (pe) parece relevante, aunque puede estar altamente relacionada con la anemia. Blood urea está muy descompensada y solo aporta información en 1 de los 4 niveles. pcc (pus cell clumps) esta directamente relaciondada con pc.



	Dim 1	Dim 2	Dim 3	Dim 4	Dim 5
sg_0	0.06	0.01	0.09	0.24	0.92
sg_1	1.22	0.65	0.18	0.04	0.18
sg_2	1.36	0.02	0.53	0.01	0.00
sg_3	0.52	0.06	0.01	0.01	2.27
sg_4	1.81	0.16	1.12	0.01	1.89
al_0	2.91	0.14	0.31	0.14	0.22
al_1	0.23	2.25	0.01	1.16	0.00
al_2	1.60	0.07	0.02	1.07	0.01
al_3	1.43	0.04	0.60	0.20	1.36
al_4	1.13	0.02	0.34	1.35	0.07

Ilustración 6. Peso de cada variable en la segunda dimensión, resumen del peso de algunas variables en las 5 primeras dimensiones.

En la segunda dimensión apenas hay variables relevantes destacan: pot (potasio), so (sodio), sc (serum creatine), y bu (blood urea). la variable pot ha de ser eliminada ya que representa muy poca variabilidad de los 200 pacientes 197 pertenecen al nivel 0 de la variable. No es conveniente mantener una variable tan sesgada.

Variable	Niveles			Muestras por nivel		
Blood urea (bu)	0	3	6	108	5	1
	1	4	7	16	5	1
	2	5		11	1	
Sodium (sod)	0	3	6	4	14	22
	1	4	7	3	92	9
	2	5	8	6	49	1
Serum creatinine (sc)	0	3	6	159	22	1
	1	4		4	9	
	2	5		1	4	
Potassium (pot)	0	2		197	1	
	1	3		1	1	
Sugar (su)	0	3		170	8	
	1	4		6	6	
	2	5		9	1	

Tabla 1. Número de muestras en cada nivel de algunas de las variables más importantes de la segunda dimensión.

Tras el análisis MCA decidimos eliminar las siguientes variables:

- Sugar (su) y glucosa (bgr). Ambas variables tienen un peso mínimo en ambas dimensiones en todos sus niveles.
- Albumina (al). solo aparecen los niveles 0, 2, 3 y 4 entre las variables con pesos significativos además está muy descompensada.
- Rbcc (red blood cell volume) y hemo (hemoglobine), están muy relacionadas con rbc, no todos sus niveles son representativos y tienen menos peso que rbc.
- pcc (pus cell clums) y ba (bacteria) están directamente relacionadas con pc (pus cells) y aportan menos información.
- pot (potasium) de 200 muestras 197 pertenecen al nivel 0.
- pcv (packed cell volume), wbcc (white blood cell counts), sg (specific gravity), bu (blood urea), sod.
- (sodium) y age. pocos o ninguno de los niveles aparecen en el top 48 variables significativas.

En resumen, mantendremos las siguientes variables:

- rbc (red blood cells).
- pc (pus cells).
- htn (hypertension).
- cad (coronary artery disease).
- dm (diabetes melitus).
- pe (pedal anemia).
- appet (apetito).
- Ane (anemia).

Como sospechábamos que podría haber cierta colinealidad entre las variables, rbc, ane y pe además de entre htn y cad realizamos un test de multicolinealidad con la función *`assoc`* del paquete vcd.

El resultado obtenido fue el siguiente:

```
                    rbc  0                                   1
                    htn  0                   1              0              1
                    cad  0       1           0       1      0       1      0       1
                    ane  0   1   0   1   0   1   0   1   0   1   0   1   0   1   0   1
affected pc dm appet
0        0  0  0         72   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
                1          0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
            1   0          0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
                1          0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
         1  0   0          0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
                1          0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
            1   0          0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
                1          0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
1        0  0  0         16   0   0   1   7   0   2   0   2   1   0   0   2   0   0   1
                1          2   2   0   0   1   0   1   0   0   0   0   0   0   0   0   0
            1   0          6   1   0   0  14   4   3   2   1   0   0   0   2   0   0   0
                1          2   0   2   0   5   1   2   0   0   0   0   0   0   0   0   0
         1  0   0          4   1   0   0   0   1   1   0   1   0   0   0   1   1   0   0
                1          2   2   0   0   1   2   0   0   1   1   0   0   1   0   0   0
            1   0          2   0   0   0   2   4   1   1   0   0   0   0   1   0   2   0
                1          0   0   0   0   3   1   1   0   0   0   0   0   1   4   1   1

  0   1
 72 128
```

Ilustración 7. Tabla de contingencia entre las variables elegidas y la variable respuesta.

Esta tabla permite observar que hay una clara relación entre todas estas variables y la respuesta. Ninguno de los pacientes sanos mostró ninguno de los síntomas.

Si un paciente sufre de falta de apetito, anemia, hipertensión etc. Para nuestro modelo va a significar que padece CKD lo cual es un error. Por tanto, no podemos incluir ninguna de estas variables y debemos pensar en añadir alguna de las descartadas anteriormente.

Si creamos el modelo con algunas de estas variables, por ejemplo, el apetito y la hipertensión recibiremos el siguiente aviso. *"Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred".*

```
Call:
glm(formula = affected ~ appet + htn, family = binomial(link = "logit"),
    data = data_reduced)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-0.90052  -0.90052   0.00007   0.00008   1.48230

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -0.6931     0.2041  -3.396 0.000684 ***
appet1       19.7208  2195.7879   0.009 0.992834
htn1         20.6505  1813.5502   0.011 0.990915
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 261.37  on 199  degrees of freedom
Residual deviance: 137.49  on 197  degrees of freedom
AIC: 143.49
```

Ilustración 8. Resumen del modelo usando como variables predictoras la hipertensión y falta de apetito.

Este mensaje es un aviso que nos indica que las probabilidades ajustadas del modelo son muy cercanas a 0 o 1. Esto puede suceder si el modelo está sobreajustado a los datos o si hay variables predictoras altamente correlacionadas o redundantes. También puede suceder si hay valores atípicos extremos en los datos o si la distribución de los datos es muy desequilibrada. En mi opinión la base de datos parece sesgada, no es normal que toda la gente que muestra falta de apetito padezca de CKD y es un resultado que no obtendríamos si preguntásemos a la gente en la calle.

Si nos fijamos en los errores estándar estos son altísimos y las variables no resultan significativas. Lo mismo pasa si añadimos todas las variables seleccionadas, vemos que "Pr(>|Z|)" es 1 o prácticamente 1, lo que sugiere que ninguna de las variables predictoras está aportando información útil al modelo. Además, los errores estándar se disparan lo que indica que la estimación del cociente no es precisa.

```
Call:
glm(formula = affected ~ ., family = binomial(link = "logit"),
    data = data_reduced)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-0.6335  -0.6335   0.0000   0.0000   1.8465

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)   -1.5041     0.2764  -5.442 5.27e-08 ***
rbc1          20.6668  6086.4915   0.003    0.997
pc1           20.7970  4639.7130   0.004    0.996
htn1          20.8902  3714.9542   0.006    0.996
dm1           20.8906  3983.7550   0.005    0.996
cad1          17.8372 13496.0331   0.001    0.999
appet1        20.2734  4984.4563   0.004    0.997
ane1           2.2490 11785.9957   0.000    1.000
---
```

Ilustración 9. Resumen del modelo que incluye todas las variables seleccionadas tras el MCA.

Una posible solución es crear una regresión lasso con glmnet.

Regresión lasso:

La regresión lasso es una técnica que se usa tanto en regresiones lineales como logísticas o de Poisson que tiene la propiedad de seleccionar automáticamente las variables predictoras más importantes, eliminando las variables predictoras irrelevantes al forzar los coeficientes correspondientes a cero. Es importante destacar que para poder realizar este análisis debemos convertir las variables a codificación one-hot. La codificación "one-hot" es una técnica de codificación utilizada en el aprendizaje automático y la minería de datos para representar variables categóricas como variables numéricas.

Por ejemplo, si se tiene una variable categórica "color" con tres posibles valores: rojo, verde y azul, se crearían tres variables binarias: "color_rojo", "color_verde" y "color_azul". Para cada fila en los datos, la variable binaria correspondiente al valor real de la variable categórica tendría el valor 1 y las demás variables binarias tendrían el valor 0.

affected	rbc_0	rbc_1	pc_0	pc_1	htn_0	htn_1	dm_0
1	1	0	1	0	1	0	1
1	1	0	1	0	1	0	1
1	0	1	0	1	1	0	1
1	1	0	1	0	1	0	1
1	1	0	1	0	1	0	0

Tabla 2. Estructura de la base de datos tras la codificación one-hot.

Tras realizar el análisis se obtuvieron los siguientes resultados:

```
15 x 1 sparse Matrix of class "dgCMatrix"
                      s1
(Intercept)  1.426985e+01
rbc_0       -2.250223e+00
rbc_1        3.340790e-14
pc_0        -2.779686e+00
pc_1         5.514804e-15
htn_0       -3.447795e+00
htn_1        1.085035e-16
dm_0        -3.149484e+00
dm_1         1.311904e-13
cad_0       -1.577518e-01
cad_1        7.447258e-16
appet_0     -2.261993e+00
appet_1      1.433208e-15
ane_0       -1.425949e+00
ane_1        9.663469e-03
```

Ilustración 10. Coeficientes obtenidos tras realizar la regresión lasso.

Como se puede ver, todos los coeficientes son muy cercanos a cero, lo que indica que estas variables al parecer no son útiles para predecir la variable respuesta. Cuando realizamos el análisis incluyendo todas las variables el resultado que obtenemos es el mismo, ninguna de las variables parece significativa.

Análisis de componentes principales (PCA):

El análisis de componentes principales (PCA) es una técnica de reducción de dimensionalidad que se utiliza para transformar un conjunto de variables correlacionadas en un conjunto más pequeño de variables no correlacionadas llamadas "componentes principales".

La idea detrás de PCA es reducir la complejidad de un conjunto de datos al encontrar las variables que contribuyen más a su variabilidad y proyectar los datos en un espacio de menor dimensión, manteniendo la mayor cantidad posible de información. Esta es técnica se usa normalmente con variable numéricas, pero aprovechando que hemos creado un *`dataset`* nuevo con codificación one-hot podemos ver qué resultados nos otorga el análisis PCA.

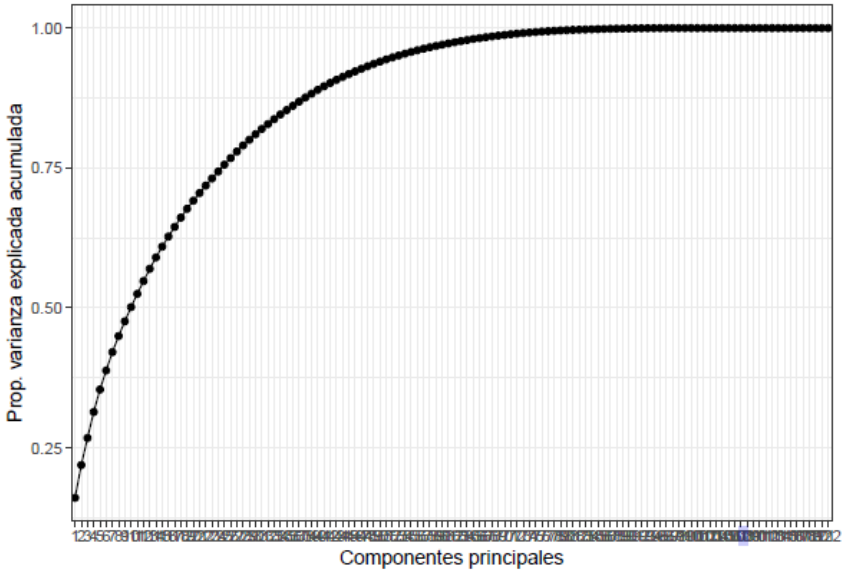


Ilustración 11. Variabilidad explicada por cada componente

El número de componentes es tan grande que no se aprecia bien cuantas componentes son necesarias en el análisis, si consultamos los valores numéricos con las 20 primeras dimensiones abarcamos un 70% de la variabilidad respecto a las 31 que necesitábamos en el MCA.

Vamos a crear un `*dataset*` con las 20 primeras componentes y a crear un modelo con ellas a ver que tal funciona.

```
Call:
glm(formula = affected ~ ., family = binomial, data = dimensiones)

Deviance Residuals:
       Min          1Q      Median          3Q         Max
-2.409e-06  -2.409e-06   2.409e-06   2.409e-06   2.409e-06

Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)      -2.657e+01  8.557e+04       0        1
`data$affected`1  5.313e+01  1.278e+05       0        1
PC1               2.232e-07  4.136e+04       0        1
PC2               1.770e-07  3.880e+04       0        1
PC3               9.876e-08  3.513e+04       0        1
PC4              -1.761e-07  3.573e+04       0        1
PC5               1.098e-07  3.910e+04       0        1
PC6               2.613e-08  4.154e+04       0        1
PC7              -1.390e-07  4.374e+04       0        1
PC8              -2.060e-07  4.681e+04       0        1
PC9               2.047e-07  4.707e+04       0        1
PC10             -2.149e-07  4.915e+04       0        1
PC11              9.578e-08  4.995e+04       0        1
PC12              3.453e-07  5.060e+04       0        1
PC13              3.186e-07  5.156e+04       0        1
PC14              1.373e-07  5.437e+04       0        1
PC15             -1.425e-07  5.928e+04       0        1
PC16             -3.923e-08  5.826e+04       0        1
PC17              9.838e-09  5.889e+04       0        1
PC18             -1.695e-07  5.918e+04       0        1
PC19              9.257e-08  6.094e+04       0        1
PC20             -9.425e-08  6.398e+04       0        1
PC21              4.361e-08  6.529e+04       0        1
```

Ilustración 12. Resume del modelo creado con las 21 primeras dimensiones.

De nuevo los errores estándar son muy altos, tenemos estimaciones cercanas a 0 valores Pr(>|Z|) iguales a 1.

En conclusión, parece que no hay ninguna combinación de variables que no resulte problemática pues la toma de datos parece bastante sesgada, sin embargo, vamos a continuar con los modelos predictivos usando las variables seleccionadas tras el análisis MCA a ver qué resultados obtenemos.

Construcción y mejora de los modelos:

EL primer paso será equilibrar la variable respuesta, inicialmente tan solo 72 muestras corresponden a la clase "nockd" codificada con un "0" y 128 la clase "cdk" codificada con un 1. El desequilibrio en la variable respuesta puede provocar que nuestros modelos tengan un sesgo hacia la clase mayoritaria, podemos resolver este problema sobremuestreando de forma sintética la clase minoritaria, para ello usamos el paquete ROSE.

Finalmente, el nuevo *`dataset`* con la adición de muestras sintéticas tiene 127 muestras "nockd" frente a 129 muestras "ckd" habiendo prácticamente una proporción del 50% entre ambas clases.

El siguiente paso es dividir el *`dataset`* en dos partes una para el entrenamiento de los modelos y otra para testear el funcionamiento de los modelos, la proporción será de un 66% de las muestras en el conjunto de entrenamiento y el 34% en el conjunto de test.

K-Nearest Neighbor (KNN):

El algoritmo funciona calculando la distancia entre el punto de datos que se está intentando clasificar y los puntos de datos más cercanos en el conjunto de entrenamiento. Luego, utiliza las etiquetas de clase de los k vecinos más cercanos para determinar la etiqueta de clase del punto de datos de prueba.

Teóricamente el mejor de los valores para k equivale a al número impar más cercano a la raíz cuadrada del número total de muestras de entrenamiento. Se busca un número impar para que las votaciones nunca empaten. En este caso el valor k óptimo parece ser 13.

```
Confusion Matrix and Statistics

          Reference
Prediction  1  0
         1 33 12
         0  0 39

               Accuracy : 0.8571
                 95% CI : (0.7638, 0.9239)
    No Information Rate : 0.6071
    P-Value [Acc > NIR] : 5.055e-07

                  Kappa : 0.7186

 Mcnemar's Test P-Value : 0.001496

            Sensitivity : 1.0000
            Specificity : 0.7647
         Pos Pred Value : 0.7333
         Neg Pred Value : 1.0000
             Prevalence : 0.3929
         Detection Rate : 0.3929
   Detection Prevalence : 0.5357
      Balanced Accuracy : 0.8824

       'Positive' Class : 1
```

Ilustración 13. Resumen de eficiencia del modelo KNN.

Se han producido un total de 12 falsos positivos frente a 0 falsos negativos, todo apunta a que el modelo a pesar de tener la variable respuesta equilibrada sigue teniendo un sesgo hacia la clase originalmente predominante. Además, el modelo seleccionado fue el de k = 5 que quizás es algo pequeño y menor del esperado. El modelo puede pecar de sobreajustarse a los datos de entrenamiento y ser menos preciso con datos diferentes.

Support Vector Machine (SVM):
El objetivo del SVM es encontrar un hiperplano en un espacio de alta dimensión que separe los datos en diferentes categorías. El hiperplano se elige de tal manera que maximiza la distancia entre los puntos de datos de ambas categorías más cercanas.

SVM puede manejar tanto variables numéricas como categóricas. En el caso de variables categóricas, es necesario convertirlas en variables numéricas antes de aplicar SVM. Una forma común de hacerlo es mediante la codificación de etiquetas o "one-hot encoding", tal y como hicimos en la regresión Lasso. Una vez transformadas las variables procedemos a aplicar el SVM.

- **Kernel lineal:**

  Los kernels lineales son una forma de función kernel utilizada en las máquinas de vectores de soporte (SVM) para separar datos en un espacio. Una ventaja de los kernels lineales es que son computacionalmente eficientes y fáciles de implementar. Sin embargo, su capacidad de separación es limitada y no funcionan bien en conjuntos de datos no lineales.

```
Confusion Matrix and Statistics

          Reference
Prediction  1  0
         1 41  0
         0  4 39

               Accuracy : 0.9524
                 95% CI : (0.8825, 0.9869)
    No Information Rate : 0.5357
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9049

 Mcnemar's Test P-Value : 0.1336

            Sensitivity : 0.9111
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.9070
             Prevalence : 0.5357
         Detection Rate : 0.4881
   Detection Prevalence : 0.4881
      Balanced Accuracy : 0.9556

       'Positive' Class : 1
```

Ilustración 14. Resumen modelo SVM con kernel lineal.

Tal como mencionábamos en la PEC 1 los SVM son generalmente buenas opciones con dataset de variables categóricas.

- **Kernel radial:**

  El kernel radial o Gaussiano se utiliza normalmente para clasificar datos no lineales en un espaico dimensional superior. Los resultados obtenidos con el kernel lineal parecen mostrar que los datos pueden clasificar en un espacio bidimensional por lo que este kernel no debería mejorar los resultados.

```
Confusion Matrix and Statistics

          Reference
Prediction  1  0
         1 41  0
         0  4 39

               Accuracy : 0.9524
                 95% CI : (0.8825, 0.9869)
    No Information Rate : 0.5357
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9049

 Mcnemar's Test P-Value : 0.1336

            Sensitivity : 0.9111
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.9070
             Prevalence : 0.5357
         Detection Rate : 0.4881
   Detection Prevalence : 0.4881
      Balanced Accuracy : 0.9556
```

Ilustración 15. Resumen del modelo SVM con kernel radial.

  El resultado obtenido es el mismo, es posible que los datos sean linealmente separables. En este caso, un kernel lineal es suficiente para encontrar la mejor separación de las clases en el espacio de características.

Decision tree:

Este método de aprendizaje automático se basa en la construcción de un árbol mediante la selección de las mejores variables para dividir los datos en cada nivel, y esta selección se basa en un criterio que busca maximizar la homogeneidad de los grupos resultantes o minimizar la impureza de estos. Esto se realiza hasta que los nodos no se pueden dividir más, y se obtiene el árbol completo.

Una de las principales ventajas de los árboles de decisión es que son fáciles de interpretar y visualizar. Además, pueden manejar tanto variables categóricas como numéricas, y son resistentes a valores atípicos y valores faltantes.

En este caso se obtuvo un número muy grande de falsos negativos que son quizá aún más preocupantes que los falsos negativos pues pueden dar lugar a diagnósticos mucho más tardíos y en fases en los que la enfermedad está demasiado avanzada. En la anterior PEC comentábamos que los árboles de decisiones son muy útiles con este tipo de datos por lo que un resultado tan malo es bastante sorprendente.

```
Confusion Matrix and Statistics

          Reference
Prediction  1  0
         1 33  0
         0 12 39

               Accuracy : 0.8571
                 95% CI : (0.7638, 0.9239)
    No Information Rate : 0.5357
    P-Value [Acc > NIR] : 4.235e-10

                  Kappa : 0.7186

 Mcnemar's Test P-Value : 0.001496

            Sensitivity : 0.7333
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.7647
             Prevalence : 0.5357
         Detection Rate : 0.3929
   Detection Prevalence : 0.3929
      Balanced Accuracy : 0.8667
```

Ilustración 16. Resumen del modelo "decision tree".

Random forest:

En lugar de construir un solo árbol de decisión, Random Forest construye una cantidad de árboles de decisión y luego realiza una votación para la clasificación final.

Vamos a crear 3 modelos, uno con 5 árboles otro con 20 otro con 40 y el último con 80 a ver cuál es el que mejor funciona. Por lo general, se recomienda usar un número suficientemente grande de árboles para que el modelo tenga una buena precisión, pero no tan grande que el costo computacional se vuelva prohibitivo. Por tanto, el modelo con mayor número de árboles debería ser el más preciso.

Para reducir la cantidad de datos mostrados en el informe solo se mostrarán las matrices de confusión de los modelos de menor a mayor número de árboles.

```
          Reference
Prediction  1  0
         1 40  0
         0  5 39
```

```
          Reference
Prediction  1  0
         1 40  0
         0  5 39
```

```
          Reference
Prediction  1  0
         1 39  0
         0  6 39
```

```
          Reference
Prediction  1  0
         1 40  0
         0  5 39
```

Ilustración 17. Matrices de confusión de cada modelo "random forest" con 5 árboles (arriba derecha), 20 árboles (abajo izquierda), 40 árboles (arriba derecha), 80 árboles (abajo derecha).

La precisión prácticamente no varía en ninguno de los casos esto puede ser debido a que con tan solo 5 árboles ya se ha alcanzado el límite de convergencia del modelo. Es decir, es posible que el modelo haya logrado la mejor combinación posible de precisión y complejidad con solo 5 árboles, y no haya necesidad de agregar más árboles para mejorar la precisión del modelo. Normalmente se alcanza el límite de convergencia con un número bajo de árboles cuando el conjunto de datos no es muy grande o cuando hay pocas variables como es este caso.

Regresión binomial:

Por último, vamos a probar con la regresión, aunque como sabemos los predictores no van a resultar significativos.

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


Call:
glm(formula = datos_train_labels ~ ., family = binomial, data = train_reg)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.2431   0.0000   0.4105   0.4105   0.4105

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)    2.4314     0.3688   6.593  4.3e-11 ***
rbc1         -22.3255  6905.5463  -0.003    0.997
pc1          -22.1714  5624.0180  -0.004    0.997
htn1         -21.9991  4702.4345  -0.005    0.996
dm1          -21.7339  5442.0485  -0.004    0.997
cad1          20.2800 12147.0380   0.002    0.999
appet1       -21.7765  7413.4117  -0.003    0.998
ane1          -0.7766  8524.2858   0.000    1.000
---
Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 237.861  on 171  degrees of freedom
Residual deviance:  55.586  on 164  degrees of freedom
AIC: 71.586

Number of Fisher Scoring iterations: 21

Warning in confusionMatrix.default(reg_pred_class, datos_test_labels): Levels
are not in the same order for reference and data. Refactoring data to match.

Confusion Matrix and Statistics

          Reference
Prediction  1  0
         1 41  0
         0  4 39
```

Ilustración 18. Resumen del modelo de regresión binomial junto con la matriz de confusión.

Los resultados obtenidos son buenos pero el modelo no parece muy fiable, los errores son altísimos y las variables no son significativas.

A pesar de que algunos de los resultados obtenidos en ciertos modelos son positivos y están por encima de las precisiones que estipulamos en la planificación del trabajo creo que el hecho de que la mayoría de los niveles de muchas variables estén muy desbalanceados están provocando problemas en el análisis. Una posible solución es agrupas los niveles minoritarios para compensar las variables teniendo en cuenta la perdida de información que esto provocaría por otro lado si las variables no son muy importantes para el modelo podemos plantear eliminarlas.

Segunda selección de variables:
Revisamos el resumen del `*dataset*` (pag 2) y buscamos que niveles de variables se pueden agrupar y que variables están tan descompensadas que no vale la pena tener en cuenta en el análisis.

Modificaciones:

- Sg: Las 3 muestras del nivel 0 se englobarán en el nivel 1.
- al: los niveles del 1 al 4 se unirán en uno solo.
- rbcc: 0:3, 4, 5:8.
- wbcc: 0:5, 6, 7:8.
- bgr: unir los niveles del 2 al 9.
- bu: Unir los niveles del 1 al 7.
- hemo: unir 0:2, 3:5 y 6:9.
- pcv: 0:3, 4:5, 6, 7:9.
- age: 0:4, 5:6, 7:9.

Del resto de variables eliminaremos:

- pot, sod, su, sc, appet, ane y pe están tan desequilibradas que no vale la pena incluirlas.
- rbc (red blood cells) ya que está muy correlacionada con rbcc (red blood cells volume).
- pcc (pus cell clums) y ba (bacteria), ambas variables se correlacionan entre sí además de con pc (pus cells) siendo pc la más equilibrada de las 3 variables.
- cad (coronary artery disease) está demasiado relacionada con la hipertensión además de estar desequilibrada.

Con la nueva codificación el `*dataset*` queda de la siguiente manera:

affected	sg_new	al_new	pc	bgr_new	bu_new	hemo_new	pcv_new	rbcc_new	wbcc_new	htn	dm	age_new
1	3	1	0	0	0	0	1	1	1	0	0	0
1	1	0	0	1	0	0	1	1	0	0	0	0
1	1	1	1	0	1	2	1	1	0	0	0	0
1	1	1	0	1	0	1	4	1	1	0	0	0
1	2	0	0	2	0	1	3	2	1	0	1	0

Tabla 3. Estructura del `dataset` tras unificar los niveles de ciertas variables.

```
affected sg_new al_new  pc      bgr_new bu_new  hemo_new pcv_new rbcc_new
0: 72    1:48   0:116   0:155   0:70    0:108   0:81     0:34    0:53
1:128    2:36   1: 84   1: 45   1:79    1: 92   1:65     1:45    1:96
         3:75                   2:51            2:54     3:56    2:51
         4:41                                            4:65
wbcc_new htn     dm      age_new
0:72     0:122   0:130   0:44
1:98     1: 78   1: 70   1:64
2:30                     2:92
```

Ilustración 19. Resumen del `dataset` tras unificar los niveles de ciertas variables.

Variable	Codificación	Valores reales
Affected	0 / 1	nockd / ckd
Specific gravity (sg_new)	1	< 1.007 - 1.011
	2	1.011 - 1.017
	3	1.019 - 1.021
	4	> 1.023
Albumin (al_new)	0 / 1	0 / 1:4
Pus cells (pc)	0 / 1	0 / 1
Blood glucosa random (bgr_new)	0	< 112
	1	112 – 154
	2	> 154
Blood urea (bu_new)	0	< 48.1
	1	> 124.3
Hemoglobin (hemo_new)	0	< 6.1 – 8.7
	1	8.7 – 12.6
	2	> 12.6
Packed cell volume	0	<17.9 – 29.6
	1	29.6 – 37.4
	3	37.4 – 41. 3
	4	> 41.3
Red blood cells volume (new_rbcc)	0	< 2.69 – 4.46
	1	4.46 - 5.05
	2	> 5.05
White blood cells volume (new_wbcc)	0	< 4980 – 16880
	1	16880 – 19260
	2	> 19260
Hypertension (htn)	0 / 1	0 / 1
Diabetes melitus	0 / 1	0 / 1
Age (age_new)	0	<12 – 43
	1	43 – 59
	2	> 59

Tabla 3. Tabla resumen de la nueva codificación.

Vamos a comprobar de forma visual como se relacionan algunas de estas variables con las enfermedad crónica de riñón. Como es de esperar a medida que avanza la edad las probabilidades de padecer CKD aumentan.

En otro gráfico vemos la relación de la hemoglobina con la variable respuesta en cada uno de los 3 niveles de edad. Este gráfico parece apuntar que niveles medios de hemoglobina en sangre están relacionados con la ausencia de CKD por otro lado cuando los niveles de hemoglobina son muy altos o en especial si son bajos parecen ser un síntoma de CKD.

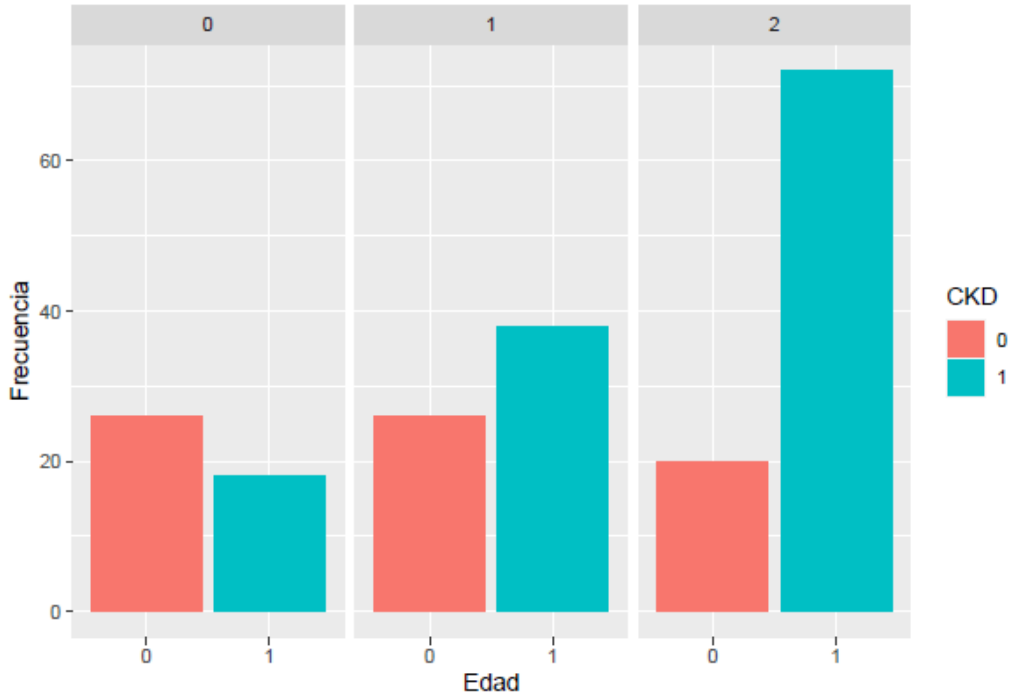


Ilustración 20. Relación entre CKD y los tres niveles de edad.

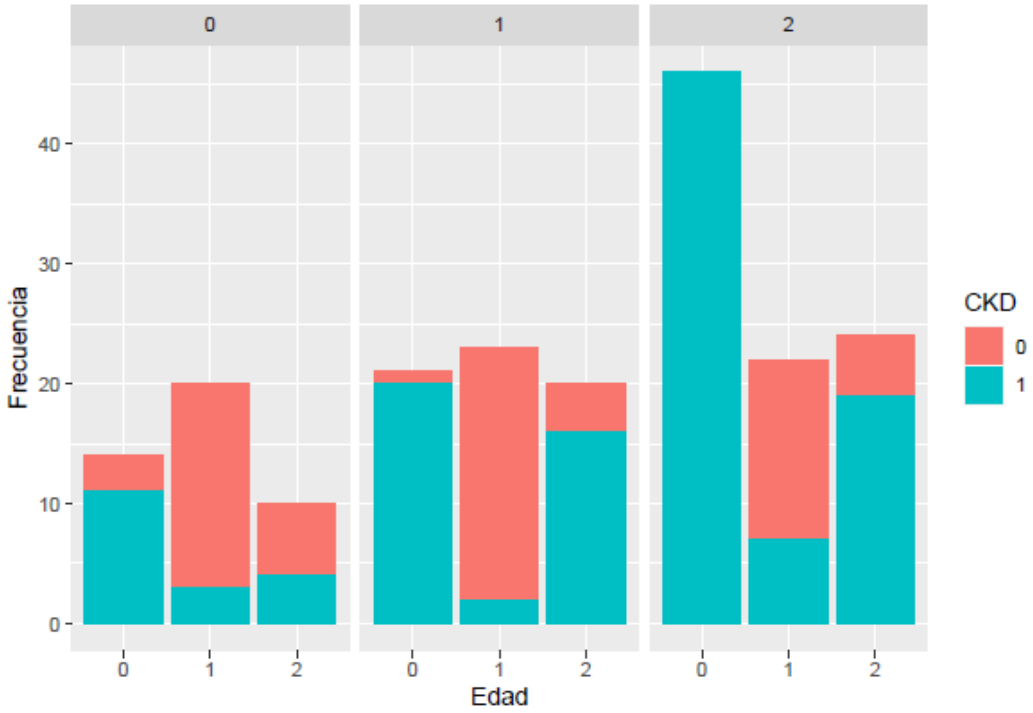


Ilustración 21. Relación entre la hemoglobina y CKD en las distintas edades.

Variable conflictivas:

Como mencinamos anteriormente hay algunas variables que tienen uno de sus niveles totalmente relacionadocon la enfermedad crónica de riñón. Por ejemplo, el 100% de los pacientes con hipertensión tenían CKD, lo mismo pasa en variables como los niveles de albumina, con la variable pc (pus cell) o dm (diabetes melitus).

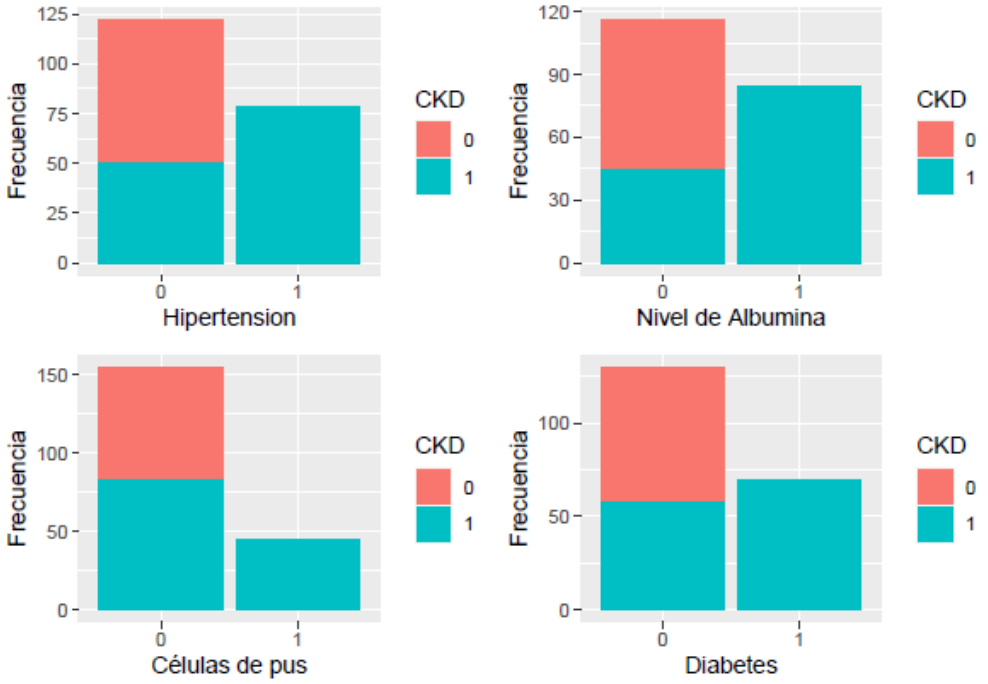


Ilustración 22. Variables en las que 1 nivel está relacionado completamente con padecer CKD.

Estas variables parecen fuertes predictores de CKD pero debemos tener en cuenta que el desbalanceo de la variable `affected` puede provocar un descenso en la precisión del modelo.

Técnicas de reducción de multidimensionalidad:
Igual que en el caso anterior empezaremos por el MCA que es el procedimiento estándar cuando se reduce la dimensionalidad de bases de datos con variables únicamente categóricas.

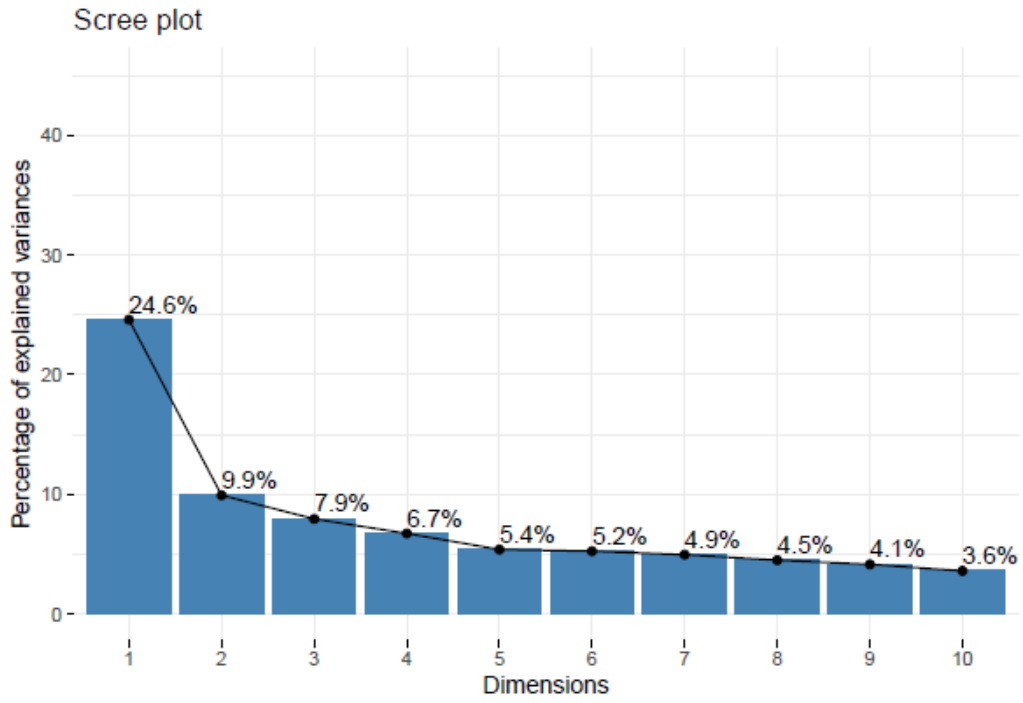


Ilustración 23. Variabilidad explicada por cada dimensión.

Vemos como con las 9 primeras dimensiones abarcamos más del 70% de la variabilidad total y en concreto con las dos primeras casi un 35 %. Este resultado es mucho mejor que el del análisis anterior donde necesitábamos 31 dimensiones para alcanzar el 70% de la variabilidad y las dos primeras solo abarcaban el 12% de la variabilidad.

Visualizamos la posición de cada variable en las dos primeras dimensiones.

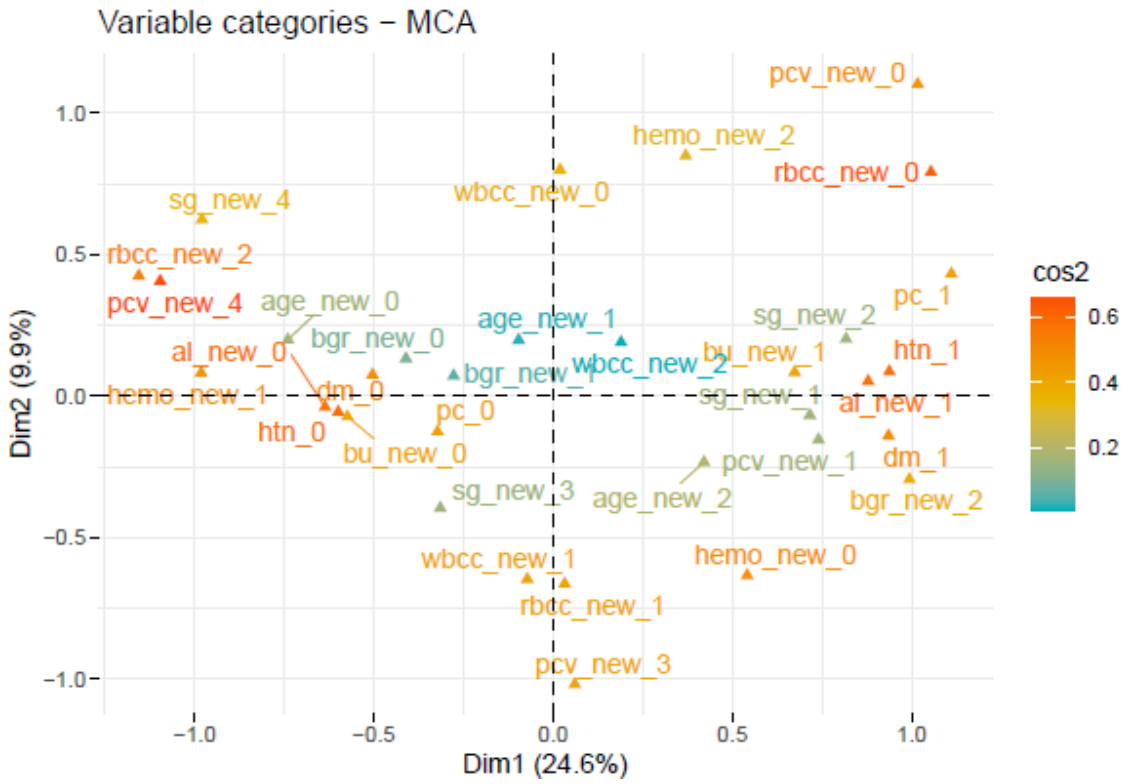


Ilustración 24. Representación de los variables sobre las dos primeras dimensiones.

Vemos como ahora las variables se distribuyen de forma más homogénea y no hay tantas variables "outliners" como pasaba en el anterior MCA.

En la primera dimensión htn en su nivel 1 tiene un valor positivo bastante alto y en su nivel 0 un valor muy negativo, otras variables como al_new, dm_new, bu_new y pc ocupan posiciones muy parecidas a htn esto puede indicar que las variables están bastante relacionadas y que aportan la misma información. Tiene sentido pensar que la existe una correlación entre la hipertensión y la diabetes. De hecho, se sabe que la hipertensión es una de las complicaciones más comunes de la diabetes y que las personas con diabetes tienen un mayor riesgo de desarrollar hipertensión.

Las variables rbcc y pcv parecen estar correlacionadas, ambas hacen referencia al volumen celular en sangre y varios de sus niveles ocupan ubicaciones parecidas en la representación.

La edad no parece poder aportar más información de la que aporta la hipertensión con la que está relacionada,

lo mismo pasa con wbcc, su aportación en la primera dimensión es casi nula a pesar de ser algo importante en la segunda.

Otra variable que aporta poca información es la variable bgr que está totalmente relacionada con la diabetes, lo cual tiene sentido pues los enfermos de diabetes pueden tener sus niveles de glucosa alterados.

Todos los niveles de la variable hemoglobina destacan en alguna dimensión o en ambas.

En conclusión, las variables hipertensión hemoglobina y rbcc parecen aportar información relevante y vale la pena mantenerlas.

Si nos fijamos donde quedan representadas las muestras:

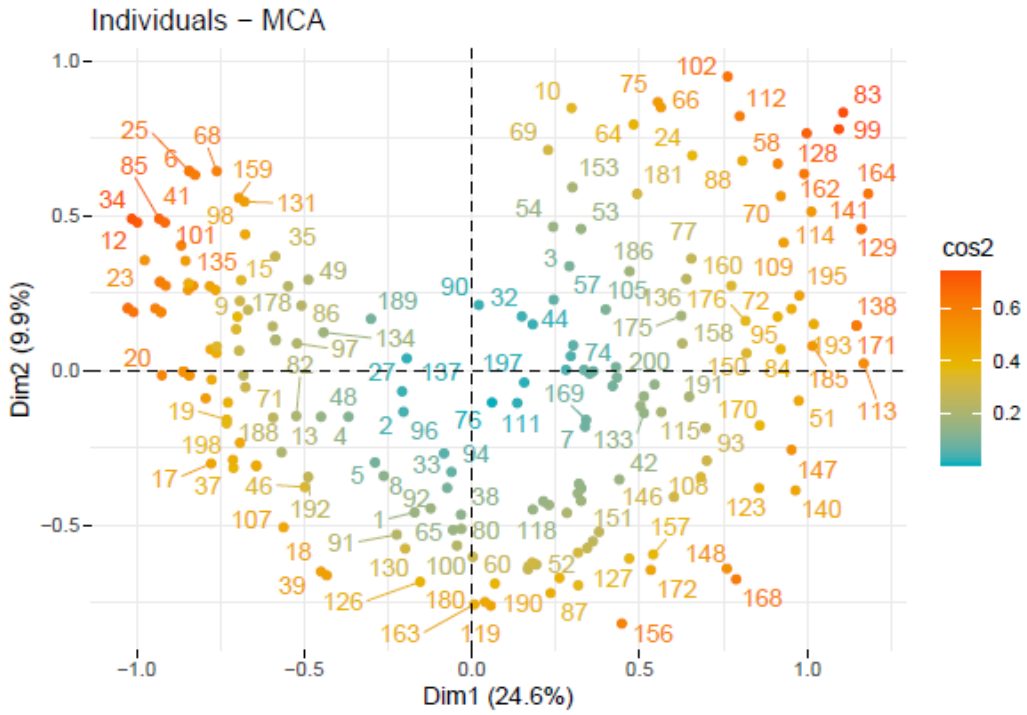


Ilustración 25. representación de los individuos sobre las dos primeras dimensiones.

La mayoría de las muestras se concentran en lugares donde aparecen niveles de rbcc htn o hemo. El análisis apunta a que podríamos lograr un buen modelo tan solo con esas tres variables.

Creamos un modelo de regresión con estas 3 variables:

```
Call:
glm(formula = affected ~ hemo_new + htn + rbcc_new, family = binomial,
    data = data_2)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.2611  -0.1008   0.0000   0.0516   3.2511

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)    3.8495     1.3307   2.893  0.00382 **
hemo_new1     -3.8816     0.8114  -4.784 1.72e-06 ***
hemo_new2     -1.8123     0.8303  -2.183  0.02905 *
htn1          21.8034  1605.4013   0.014  0.98916
rbcc_new1     -1.3740     1.2480  -1.101  0.27091
rbcc_new2     -5.2478     1.5976  -3.285  0.00102 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 261.367  on 199  degrees of freedom
Residual deviance:  65.813  on 194  degrees of freedom
AIC: 77.813
```

Ilustración 26. Resumen modelo con hemo_new, htn y rbcc_new.

El modelo parece tener un problema con la variable "htn1", ya que su coeficiente muestra un valor muy alto (21.8034) junto con un error estándar igualmente alto (1605.4013). Es posible que se deba revisar la inclusión de esta variable en el modelo y considerar si es necesario buscar más información o transformarla antes de incluirla.

Este problema puede resultar de que el 100% de los pacientes hipertensos padecen de CKD lo que puede desviar el modelo. Si en vez de la hipertensión añadimos cualquiera de las variables mencionadas anteriormente con este problema (dm, al o pc) pasa lo mismo.

Podemos comprobar que pasa si en vez de la hipertensión incluimos la urea en sangre la cual ocupa posiciones parecidas en el plano.

```
Call:
glm(formula = affected ~ hemo_new + bu_new + rbcc_new, family = binomial,
    data = data_2)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.3277  -0.2125   0.1681   0.4349   2.7577

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)   4.2638     1.1845   3.600 0.000319 ***
hemo_new1    -3.5536     0.7012  -5.068 4.03e-07 ***
hemo_new2    -2.2245     0.7660  -2.904 0.003683 **
bu_new1       1.9299     0.6041   3.194 0.001401 **
rbcc_new1    -1.9528     1.0999  -1.775 0.075826 .
rbcc_new2    -4.4900     1.2200  -3.680 0.000233 ***
---
```

Ilustración 27. Modelo con hemo_new, bu_new y rbcc_new.

Vemos que en este caso el modelo parece mucho más fiable todas las variables resultan ser significativas y tanto las estimaciones como los errores estándar parecen tomar valores fiables, aún debemos crear los modelos, pero estas 3 variables parecen bastante prometedoras a la hora de diagnosticar CKD. Los valores de VIF muestran que no parece haber ningún tipo de colinealidad entre variables contrariamente a lo que conceptualmente explican los niveles de hemoglobina y de glóbulos rojos en sangre.

```
            GVIF Df GVIF^(1/(2*Df))
hemo_new 1.086914  2        1.021054
bu_new   1.086984  1        1.042585
rbcc_new 1.055222  2        1.013529
```

Ilustración 28. Valores VIF del modelo.

Vamos a comprobar si se puede añadir alguna variable más que pueda afectar positivamente al modelo. Añadimos wbcc_new que como vimos en el MCA ya que a pesar de no aportar mucho a la primera dimensión si lo hace a la segunda.

```
Call:
glm(formula = affected ~ hemo_new + bu_new + wbcc_new + rbcc_new,
    family = binomial, data = data_2)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.6034  -0.2680   0.1467   0.3623   3.0253

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)   3.7393     1.2204   3.064 0.002184 **
hemo_new1    -3.4774     0.7102  -4.896 9.76e-07 ***
hemo_new2    -1.8944     0.7965  -2.378 0.017387 *
bu_new1       1.8357     0.6227   2.948 0.003198 **
wbcc_new1     1.2569     0.6540   1.922 0.054627 .
wbcc_new2     0.6360     0.8946   0.711 0.477165
rbcc_new1    -2.3053     1.1309  -2.038 0.041507 *
rbcc_new2    -4.8279     1.2573  -3.840 0.000123 ***
---
Analysis of Deviance Table

Model 1: affected ~ hemo_new + bu_new + rbcc_new
Model 2: affected ~ hemo_new + bu_new + wbcc_new + rbcc_new
  Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1       194     93.908
2       192     90.052  2   3.8569   0.1454
```

Ilustración 29. Resumen del modelo incluyendo wbvv más análisis anova.

El valor de Pr(>Chi) muestra el nivel de significancia para la prueba de chi-cuadrado, donde un valor menor que 0.05 indica que hay una diferencia significativa entre los modelos. En este caso, el valor de Pr(>Chi) es 0.1454, lo que indica que no hay suficiente evidencia para rechazar la hipótesis nula de que los dos modelos son iguales. Por lo tanto, no se puede concluir que agregar la variable predictora wbcc_new tenga un impacto significativo en la capacidad predictiva del modelo.

Añadir la edad tampoco parece servir. Tanto wbcc_new como age_new no aportan información significativa al modelo. Si añadimos cualquier variable distinta a estas las regresiones muestran errores estándar y estimaciones altísimas, lo mejor será quedarnos solo con "hemo", "bu_new" y "rbcc_new".

Segunda construcción y mejora de modelos:

Una vez seleccionadas las variables vamos a equilibrar la variable respuesta y a dividir el *`dataset`* en un conjunto de entrenamiento y otro de test. El 66% del *`dataset`* se usará para entrenar el modelo y el otro 34% para testearlo igual que anteriormente.

KNN:

Los valores de precisión obtenidos son altos lo que sugieren que el modelo es bastante preciso en sus predicciones con tan solo 2 falsos positivos y 2 falsos negativos.

El valor de K escogido por el modelo fue de K = 17, un poco superior al que pensábamos que era el ideal (13), por otro lado, los resultados obtenidos en este caso son mucho mejores que con las variables anteriores.

```
Confusion Matrix and Statistics

          Reference
Prediction  1  0
         1 43  2
         0  2 37

               Accuracy : 0.9524
                 95% CI : (0.8825, 0.9869
    No Information Rate : 0.5357
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9043

17-nearest neighbor model
Training set outcome distribution:

 1  0
81 91
```

Ilustración 30. Matriz de confusión del modelo KNN con 3 variables.

SVM:
Transformamos de nuevo las variable a codificación one-hot y creamos los modelos.

```
Confusion Matrix and Statistics

          Reference
Prediction  1  0
         1 41  1
         0  4 38

               Accuracy : 0.9405
                 95% CI : (0.8665, 0.9804)
    No Information Rate : 0.5357
    P-Value [Acc > NIR] : 2.762e-16

                  Kappa : 0.881

 Mcnemar's Test P-Value : 0.3711
```

```
Confusion Matrix and Statistics

          Reference
Prediction  1  0
         1 41  1
         0  4 38

               Accuracy : 0.9405
                 95% CI : (0.8665, 0.9804)
    No Information Rate : 0.5357
    P-Value [Acc > NIR] : 2.762e-16

                  Kappa : 0.881

 Mcnemar's Test P-Value : 0.3711
```

Ilustración 31. Resultados de los SVM con kernel lineal (izquierda) y radial (derecha).

El resultado obtenido con ambos modelos es el mismo, es posible que los datos sean linealmente separables. En este caso, un kernel lineal es suficiente para encontrar la mejor separación de las clases en el espacio de características.

A pesar de que los resultados son buenos tanto en este SVM como en el SVM anterior los p-valores de Mcnemar son en todos los casos mayor a 0.05 por lo que no tenemos evidencia significativa para rechazar la hipótesis nula de que ambas poblaciones son iguales.

Decision tree:
En este caso, la matriz de confusión muestra que el modelo predijo correctamente 80 de las 84 observaciones en el set de prueba, lo que resulta en una tasa de precisión del 95.24 %. El modelo tiene una sensibilidad del 91.11% y una especificidad del 100 %, lo que significa que el modelo tiene una alta capacidad para identificar tanto verdaderos positivos como verdaderos negativos.

El árbol se divide en cinco nodos. El nodo raíz (nodo 1) divide el conjunto de datos de entrenamiento en dos grupos, uno con 97 observaciones y otro con 75 observaciones. Las divisiones se basan en las tres variables predictoras. El nodo 2 tiene 97 observaciones y se

divide en dos subnodos. El nodo 3 tiene 75 observaciones, el nodo 4 tiene 73 observaciones y el nodo 5 tiene 24 observaciones.

En este caso, la tasa de error esperada para el nodo raíz (0.6296296) es relativamente alta, lo que sugiere que el modelo no es muy preciso en su predicción. Sin embargo, la tasa de error se reduce en los nodos 2 y 3, lo que indica que el modelo está logrando discriminar entre las clases en esos nodos. La proporción de observaciones en cada nodo también parece estar razonablemente balanceada.

```
Confusion Matrix and Statistics

          Reference
Prediction  1  0
         1 41  0
         0  4 39

               Accuracy : 0.9524
                 95% CI : (0.8825, 0.9869)
    No Information Rate : 0.5357
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9049

 Mcnemar's Test P-Value : 0.1336
```

Ilustración 32. Resumen modelo decisión tree.

Random forest:
Creamos de nuevo 4 modelos, uno con 5 árboles, otro con 20, otro con 40 y otro con 80.

```
##           Reference
## Prediction  1  0
##          1 40  0
##          0  5 39
```

```
##           Reference
## Prediction  1  0
##          1 43  2
##          0  2 37
```

```
##           Reference
## Prediction  1  0
##          1 41  1
##          0  4 38
```

```
##           Reference
## Prediction  1  0
##          1 41  1
##          0  4 38
```

Ilustración 33. Matrices de confusión de cada modelo "decision tree" con 5 árboles (arriba derecha), 20 árboles (abajo izquierda), 40 árboles (arriba derecha), 80 árboles (abajo derecha).

Al final se obtuvo la mejor precisión con 40 árboles, puede que en ese valor se alcanzase el límite de la mejora en la precisión del modelo aumentar por tanto el número de árboles no hará más que sumar a la carga computacional. Por otro lado, es algo extraño que en este caso se necesiten más árboles para llegar al modelo óptimo habiendo menos variables que en el caso anterior.

Regresión binomial y lasso:

```
Call:
glm(formula = datos_train_labels ~ ., family = binomial, data = train_reg)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.6944  -0.1224   0.2318   0.4485   2.3291

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  -20.4234  1666.9812  -0.012 0.990225
bu_new1       -2.2463     0.6614  -3.396 0.000684 ***
hemo_new1      4.8063     1.1697   4.109 3.97e-05 ***
hemo_new2      3.4492     1.1996   2.875 0.004036 **
rbcc_new1     17.7797  1666.9809   0.011 0.991490
rbcc_new2     19.2203  1666.9810   0.012 0.990801
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 237.861  on 171  degrees of freedom
Residual deviance:  70.787  on 166  degrees of freedom
AIC: 82.787

Number of Fisher Scoring iterations: 18
```

Ilustración 34. Resumen del modelo de regresión.

Vemos que al dividir el dataset la cantidad de datos disponibles para el análisis ha disminuido y por tanto la potencia estadística. Una forma de abordar esto es utilizando técnicas de regularización para reducir la complejidad del modelo y evitar el sobreajuste, lo que puede mejorar su capacidad de generalización. Una técnica comúnmente es la regresión logística Lasso.

```
Confusion Matrix and Statistics

          Reference
Prediction  1  0
         1 34  7
         0 11 32

               Accuracy : 0.7857
                 95% CI : (0.6826, 0.8678)
    No Information Rate : 0.5357
    P-Value [Acc > NIR] : 1.76e-06

                  Kappa : 0.5722

 Mcnemar's Test P-Value : 0.4795
```

```
7 x 1 sparse Matrix of class "dgCMatrix"
                      s0
(Intercept) -7.531181
(Intercept)  .
bu_new1     14.637285
hemo_new1    .
hemo_new2    .
rbcc_new1    .
rbcc_new2    .
```

Ilustración 35. Resumen de la regresión lasso.

La regresión lasso no ha hecho muy buenas predicciones, si nos fijamos en el peso que se le ha dado a cada variable, las variables predictoras (hemo_new1, hemo_new2, rbcc_new1, rbcc_new2) tienen coeficientes igual a cero, lo que indica que el modelo Lasso las ha eliminado ya que no eran suficientemente relevantes para la predicción de la variable respuesta. El modelo solo está teniendo en cuenta una de las variables de ahí que sea tan malo.

Resultados obtenidos:

Para comparar los modelos, compararemos las medidas de exactitud, el valor kappa, la sensibilidad y especificidad de todos los modelos utilizados tanto los primeros en los que usamos 7 variables como los segundos donde usamos solamente 3.

Modelo	Nº de Variables	Precisión	Kappa	Sensibilidad	Especifidad
KNN	7	0.86	0.72	1	0.76
SVM lineal	7	0.95	0.9	0.91	1
SVM radial	7	0.95	0.9	0.91	1
Decision Tree	7	0.86	0.72	0.73	1
Random forest (5)	7	0.94	0.88	0.89	1
Random forest (20)	7	0.93	0.86	0.87	1
Random forest (40)	7	0.94	0.88	0.89	1
Random forest (80)	7	0.94	0.88	0.89	1
Regresión binomial	7	0.95	0.9	0.91	1
KNN	3	0.95	0.9	0.96	0.95
SVM lineal	3	0.94	0.88	0.91	0.97
SVM radial	3	0.94	0.88	0.91	0.91
Decision Tree	3	0.95	0.9	0.91	1
Random forest (5)	3	0.94	0.88	0.89	1
Random forest (20)	3	0.94	0.88	0.91	0.97
Random forest (40)	3	0.95	0.9	0.96	0.95
Random forest (80)	3	0.94	0.88	0.91	0.97
Regresion Lasso	3	0.79	0.57	0.76	0.82

Tabla 4. Resumen de todos los modelos

Entre todos los modelos podemos destacar el SVM lineal con 7 variables, el KNN con 3 variables o el random forest (40) con 3 variables. Finalmente creo que la mejor opción es el KNN con 3 variables. El SVM lineal no me acaba de convencer ya que de las variables escogidas algunas están muy sesgadas y respecto al random forest me parece extraño que se requieran tantos árboles para llegar a un modelo óptimo utilizando tan solo 3 variables.

En conclusión, me decanto por el modelo KNN con 3 variables porque las variables escogidas parecen las mejores y por qué es un modelo fácil de implementar y que es óptimo para conjuntos de datos con pocas muestras o variables como es el caso.

Grado de cumplimiento de los objetivos:

A. Objetico general:

1. Crear una aplicación capaz de diagnosticar o descartar ERC utilizando únicamente algunas de las variables presentes en la base de datos.

Por ahora no hemos empezado a crear la aplicación, pero ya hemos seleccionado las variables clave para el diagnóstico de la enfermedad.

B. Objetivos específicos:

1. Evaluar distintas técnicas de clasificación para determinar el mejor modelo predictivo.
2. Optimizar el modelo seleccionado hasta lograr una precisión mínima del 85%.

Ambos objetivos están cumplidos, el segundo con creces ya que hemos llegado a una precisión un 10% superior al mínimo estipulado.

3. Encontrar las variables con mayor peso en el diagnóstico de la enfermedad y utilizarlas para crear un nuevo modelo que alcance como mínimo una precisión del 75%.

Este objetivo ha cambiado respecto a la PEC 1, obligatoriamente era necesario disminuir el número de variables ya que usando todas los modelos daban precisiones del 100% lo cual suele indicar problemas con los modelos, realmente solo es necesario quedarnos con el mejor modelo el cual ya tiene el número de variables reducidas.

4. Desarrollar una aplicación web interactiva en base al modelo con menor número de variables.

Este objetivo se desarrollará en la siguiente PEC.

Relación de las actividades realizadas:

1- Actividades previstas en el plan de trabajo:

Se realizó un análisis exploratorio de los datos, se buscaron las variables más importantes y se construyeron todos los modelos previstos.

2- Actividades no previstas y realizadas:

Cuando hablábamos de buscar las variables más importantes no especificamos ninguna técnica de reducción de dimensionalidad. Al final se aplicaron tanto MCA, como PCA como regresión Lasso.

Unificación de niveles en ciertas variables: Al ver que con las variables tal cual estaban no encontrábamos una combinación adecuada de variables nos vimos obligados a fusionar algunos niveles para nivelar las variables, este paso no estaba contemplado y además nos obligo a repetir el análisis MCA y los modelos.

En vez de una regresión binomial se tuvo que realizar una regresión lasso ya que al dividir la base de datos en entrenamiento y set la regresión perdió poder estadístico.

Desviaciones en la temporización:

En esta PEC ocurrieron varias desviaciones:

- El primer problema era que estaba obteniendo modelos perfectos todo el rato ya que estábamos introduciendo demasiadas variables, la solución fue invertir varios días en entender el análisis MCA y aplicar el resto de las técnicas.
- El segundo problema fue que aun reduciendo considerablemente el número de variables la regresiones lineales seguían dando problemas como desviaciones muy altas etc. La solución fue unificar niveles de variables y repetir todo el proceso de reducción de variables y construcción de modelos. El proceso en sí no llevo mucho tiempo ya que los códigos eran prácticamente los mismos pero el dar con la solución retrasó mucho la PEC.

Finalmente se conseguirá entregar la PEC dentro del plazo de entrega oficial pero dos días más tarde del plazo interno que estipulamos en la planificación.

No procede ningún cambio en la organización de la siguiente PEC ya que tenemos prevista empezarla el día 27 de abril y a pesar de haber tenido problemas en esta, se ha cumplido todo lo previsto, por tanto los retrasos en esta PEC no van a afectar a la siguiente.

Comentarios del tutor:

Sobre esta PEC fue la tutora quién me dio la idea de aplicar técnicas de reducción de dimensionalidad per evitar el problema de los modelos perfecto. También me dio la idea de unificar ciertos niveles de algunas variables para equilibrarlas y que de esta manera la reducción de dimensionalidad fuera mejor.

Sobre la PEC anterior.

Se me comento que la distinción entre el objetivo específico 2 y 3 no tenía sentido pues íbamos a tener que reducir la dimensionalidad igual y realmente el mejor modelo debería ser el que mejor precisión alcanzase que al final ha sido lo que ha sucedido.

Se me comentó que debía indicar mejor la fuente original de los datos. Realmente se adjuntó el link correcto, pero el hipervínculo se creó en las palabras "UCI Machine Learning Repository" lo que puede llevar a confusión.

Por último, en la PEC anterior no hable del lugar donde se implementará la aplicación. Este lugar será en GitHub, GitHub es una plataforma de desarrollo colaborativo de software que permite alojar y compartir proyectos de software en línea. Para implementar una aplicación en GitHub, se necesita crear un repositorio en GitHub y subir allí el código fuente de la aplicación. GitHub también ofrece herramientas para integrar y desplegar la aplicación en la nube o en servidores locales, como GitHub Pages, GitHub Actions y GitHub Packages.

Para la siguiente PEC se creará el repositorio de GitHub donde se implementará la aplicación junto al código fuente.