UNIVERSIDAD
DE GRANADA

Introducción a la Ciencia de Datos
máster ciencia de datos e ingeniería de computadores

Trabajo teórico/práctico

Análisis de datos, Regresión y Clasificación

Autor
Ignacio Vellido Expósito
ignaciove@correo.ugr.es

Escuela Técnica Superior de Ingenierías Informática y de
Telecomunicación

Curso 2020-2021

Índice

1. Técnicas de Clasificación

Como hemos visto en el apartado de EDA, tenemos un conjunto desbalanceado. Por la descripción del problema, cometemos un error mayor cuando clasificamos mal la clase Yes. Por tanto, se podría considerar penalizar más los falsos negativos, de manera que los algoritmos de clasificación intenten cometer menores errores de este tipo.

1.1. Algoritmo KNN

Recordamos los gráficos 1-1 con las clasificaciones, vistos en el EDA.

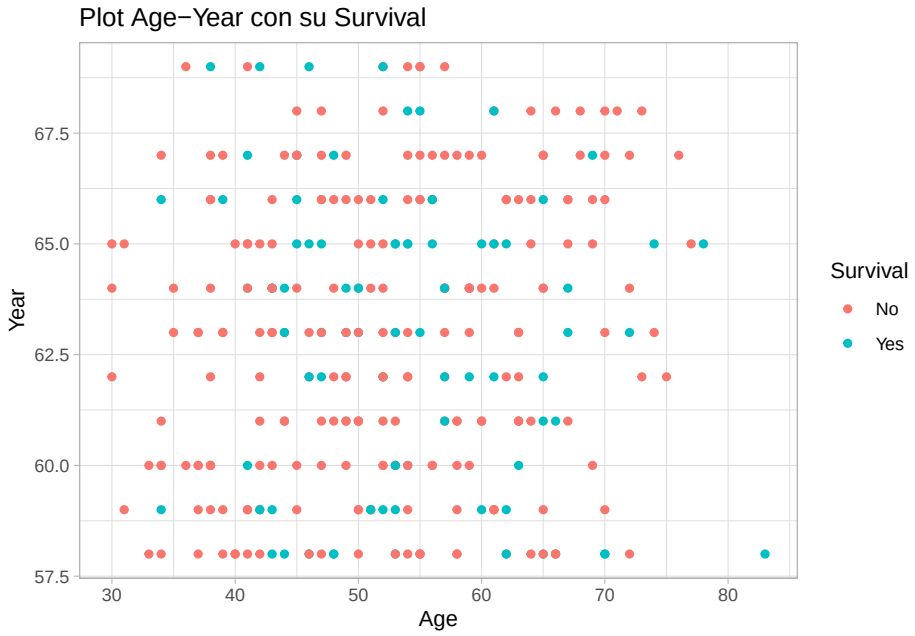

Figura 1

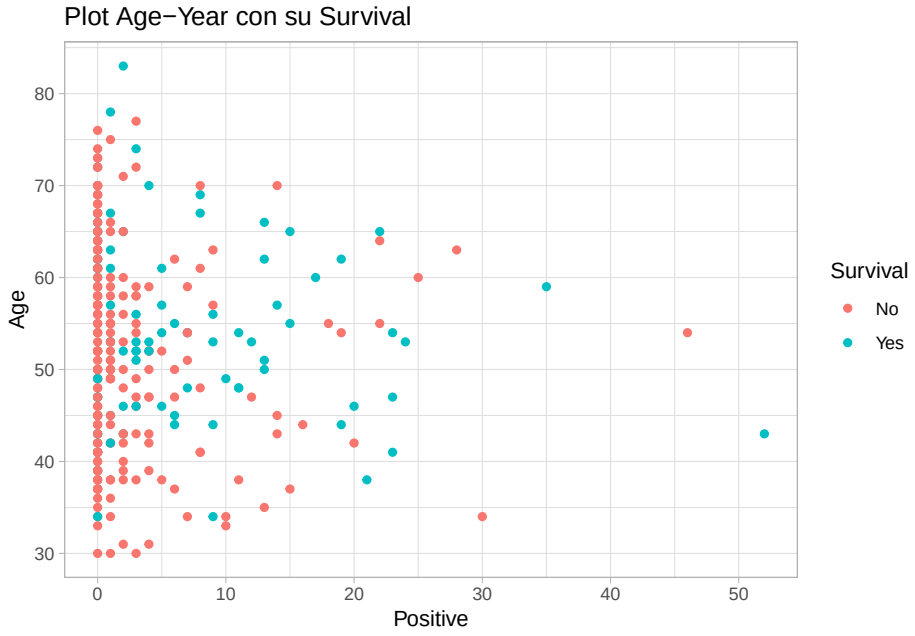

Figura 2

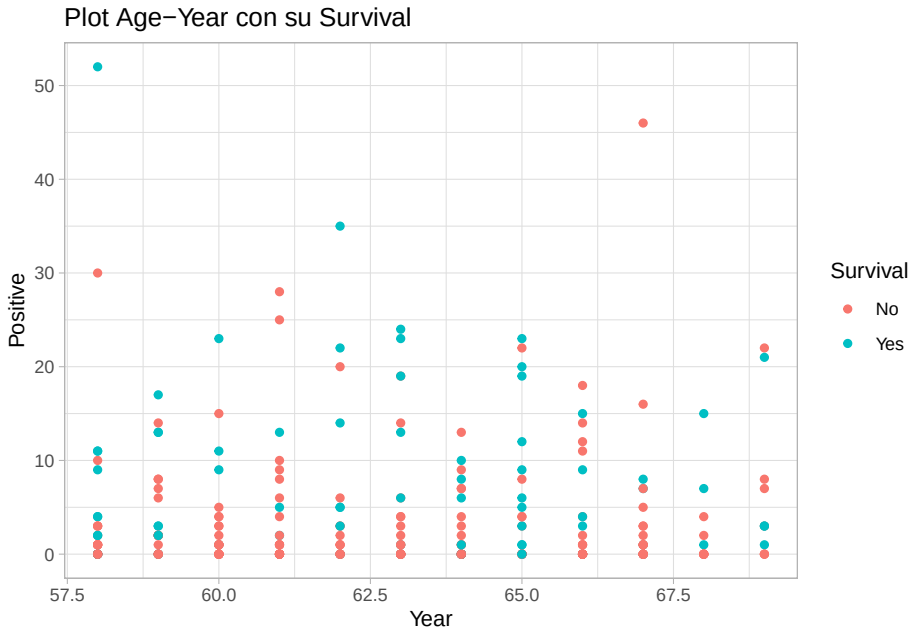

Figura 3

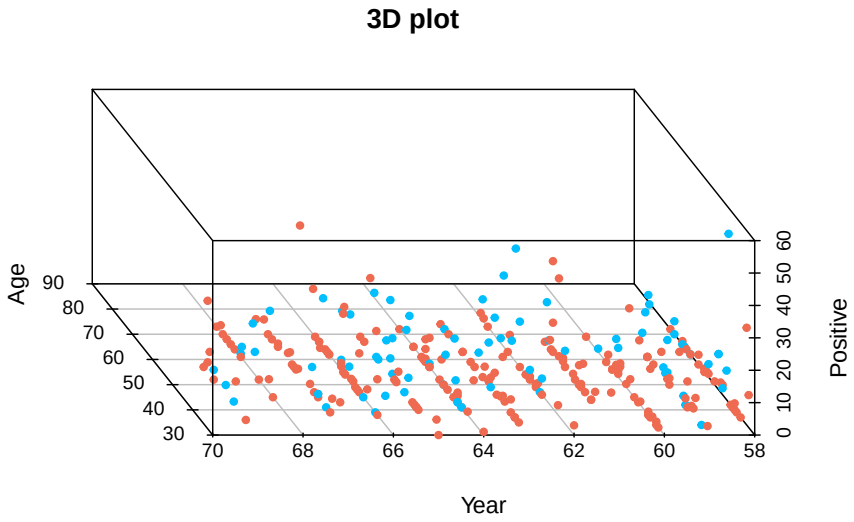

Figura 4

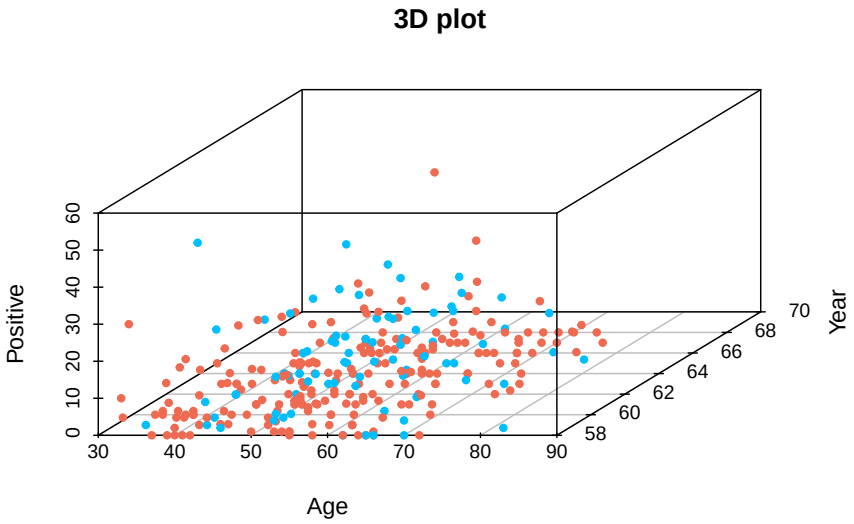

Figura 5

De cara a un algoritmo KNN, apreciamos los datos muy entremezclados, con mayor tendencia a agruparse los no supervivientes que los que sí, pero nada en especial que nos llame la atención.

Debido a esto vamos a empezar con un valor de K relativamente bajo y vamos a ir aumentándolo poco a poco. Tenemos que tener en cuenta que un K mayor puede ocasionar overfitting, pero usando técnicas de cross-validation podemos minimizar las posibilidades.

Los resultados usando el paquete caret son los siguientes[1]:

```
k-Nearest Neighbors

275 samples
  3 predictor
  2 classes: 'No', 'Yes'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 248, 248, 247, 247, 248, 247, ...
Resampling results across tuning parameters:

  k   Accuracy   Kappa
   3  0.6466931  0.05241865
   4  0.6612434  0.07153373
   5  0.6727513  0.05451908
   6  0.6907407  0.08157828
   7  0.6907407  0.07832535
   8  0.7017196  0.08311583
   9  0.7164021  0.13328924
  10  0.7089947  0.12077424
  11  0.7236772  0.15850208
  12  0.7161376  0.14004893
  13  0.7269841  0.16958772
  14  0.7197090  0.14121236
  15  0.7271164  0.16518141

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 15.
```

Vemos que al estar los datos tan entremezclados ni siquiera con un K pequeño aprende bien, es ya con un K medianamente alto (= 15) donde obtiene mayor accuracy en training.

Una vez más probablemente esto se deba a la gran mezcla de los datos, de forma que necesite la "opinión" de un gran número de vecinos para poder predecir con mayor confianza el nuevo valor.

[1] Los datos han sido preprocesados con una estandarización antes de aplicar cualquiera de los algoritmos

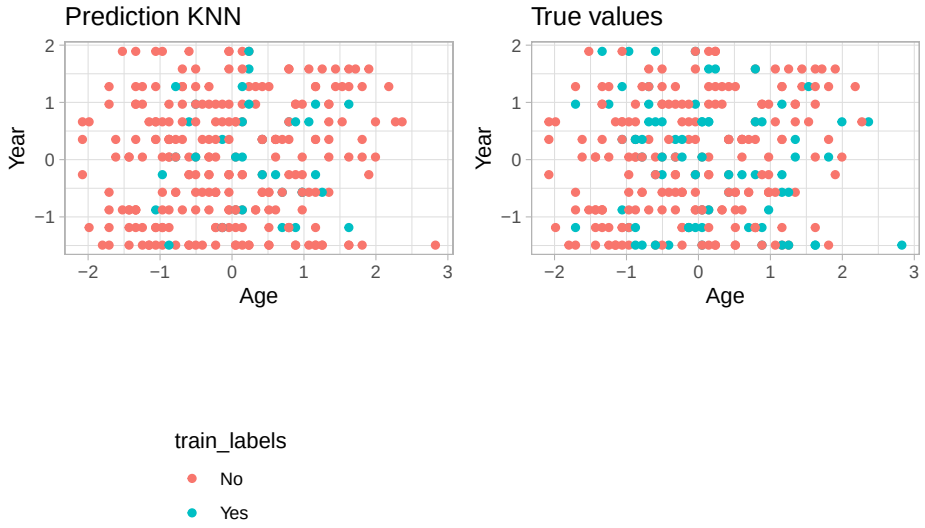

Figura 6: Predicción con K=15 en training

Vemos que el uso de un K alto hace que perdamos puntos de Yes, puesto que no hay suficientes vecinos de la misma clase que refuercen la opinión sobre esa zona del espacio.

Una evaluación con el subconjunto reservado inicialmente como test nos muestra una calidad extrañamente superior que la de training.

```
Test evaluation:
 Accuracy     Kappa
0.8387097 0.2439024

Confusion matrix (in test split):
knnPred No Yes
    No  25   5
    Yes  0   1
```

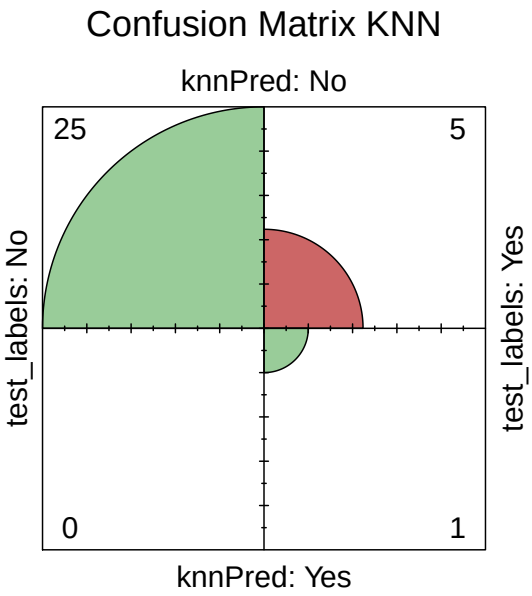

Figura 7: Matriz de confusión sobre el conjunto de test

Este comportamiento no es el habitual en aprendizaje automático, parece que casualmente el conjunto de test es bastante fácil de ajustar y por eso se obtienen mejores resultados que en training. En la sección 4.4.1 se muestran las etiquetas. También se vuelve a incidir en la alta presencia de etiquetas *No*.

El valor alto de K hace que se prediga con mayor facilidad este valor de etiqueta y por ese desbalanceo se obtengan tan buenos resultados. En sí es un poco preocupante de cara a la población real de los datos, pero debemos suponer que la muestra que tenemos es representativa y por tanto válida.

Por otro lado, para el problema que nos atañe quizás esto podría ser incluso un hecho positivo, ya que los falsos positivos sería algo que querríamos evitar a toda costa.

Por comparar, podemos también evaluar con otros valores de K en test. Puesto que hemos obtenido los mejores resultados en training con un K de 15, que es un valor relativamente alto, podemos probar con uno bajo y uno intermedio (3 y 7).

```
k-Nearest Neighbors

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 247, 248, 247, 248, 248, 247, ...
Resampling results:

  Accuracy   Kappa
  0.6654762  0.09645955

Tuning parameter 'k' was held constant at a value of 3

Test evaluation:
Accuracy     Kappa
0.8064516 0.2900763

Confusion matrix (in test split):
          test_labels
knn3Pred   No Yes
       No  23   4
       Yes  2   2
```

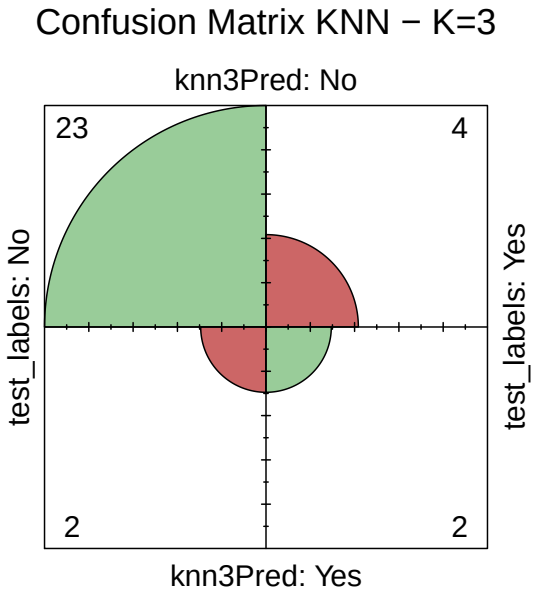

Figura 8: Matriz de confusión sobre el conjunto de test

```
k-Nearest Neighbors

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 248, 247, 247, 248, 247, 247, ...
Resampling results:

  Accuracy   Kappa
  0.6874339  0.08134908

Tuning parameter 'k' was held constant at a value of 7

Test evaluation:
Accuracy     Kappa
0.8064516 0.1696429

Confusion matrix (in test split):
        test_labels
knn7Pred No Yes
     No  24   5
     Yes  1   1
```

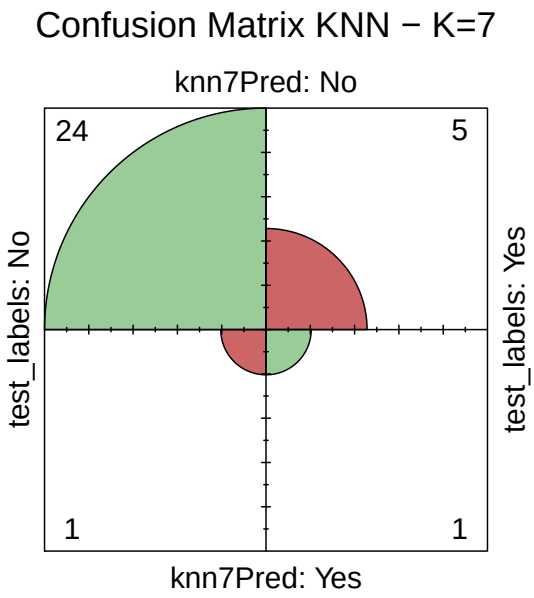

Figura 9: Matriz de confusión sobre el conjunto de test

K=7 vemos que es el que más sufre al evaluar en test, y ambos (tal y como nos había indicado la primera ejecución con CV) tienen una calidad bastante inferior (tanto en training como en test) a un K=15.

1.2. Algoritmo LDA

1.2.1. Asunciones

Comprobamos asunciones:

1. **Distribución aleatoria**: No nos queda más remedio que creer que sí.
2. **Cada predictor sigue una distribución normal**: Ya vimos en el EDA que esto no era cierto. El test de Shapiro nos aseguraba que no había normalidad y los QQ-plots nos lo hacían ver claramente. Técnicamente sabiendo esto no deberíamos usar LDA, pero puesto que esto es un proyecto seguimos.

   Por otro lado, las variables Age y Year no parecen seguir una distribución demasiado "rara" (en comparación con una normal), por lo que es posible que obtengamos resultados de calidad aceptable.
3. **Las clases siguen la misma matriz de covarianza**: Lo comprobamos a continuación.

Calculamos la diagonal de la matriz de correlación para cada una de las clases, obteniendo:

```
Para clase Yes:
      Age      Year  Positive
0.9439366 1.0656924 1.7401564

Para clase No:
      Age      Year  Positive
1.0423639 0.9998632 0.7266195
```

Estos valores nos parecen indicar que las variables Age y Positive parecen seguir distintas varianzas, pero es preferible asegurarlo con un test estadístico.

Puesto que nuestras variables no siguen una distribución normal, no podemos hacer el test de homogeneidad de Barlett. Utilizamos por tanto el de Levene

`Age:`

	Df	F value	Pr(>F)
group	1	1.799898	0.1807261
	304		

`Year:`

	Df	F value	Pr(>F)
group	1	0.0624405	0.8028481
	304		

`Positive:`

	Df	F value	Pr(>F)
group	1	18.78912	1.99e-05
	304		

Indicándonos que solo se puede asegurar que la variable Positive **no** tiene homogeneidad entre clases diferentes.

Gráficamente:

Figura 10

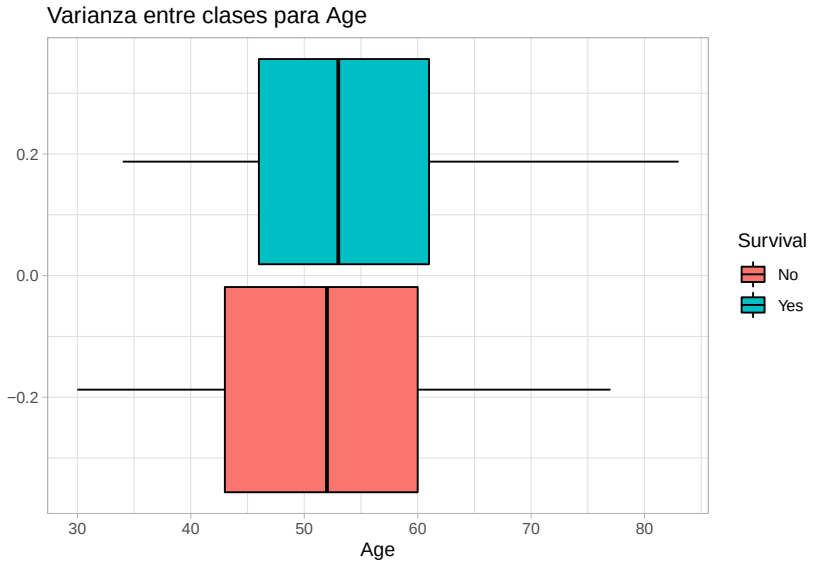

Figura 11

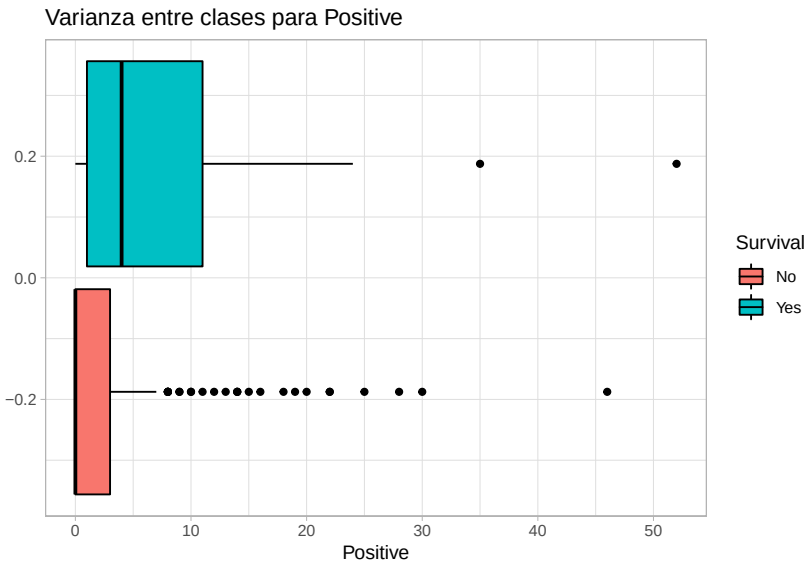

Figura 12

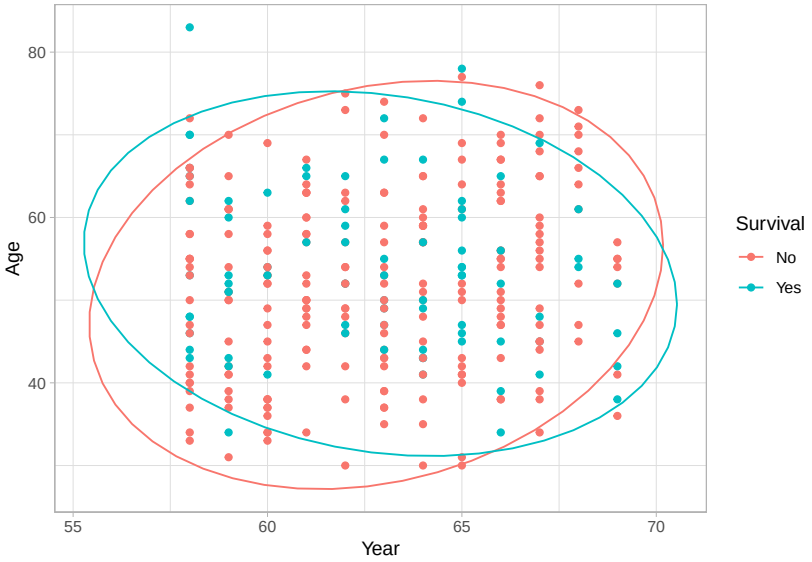

Figura 13

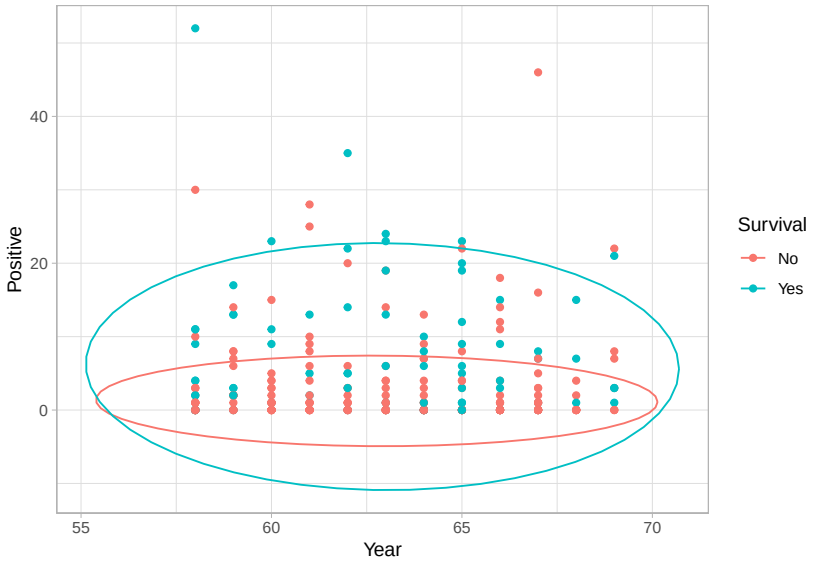

Figura 14

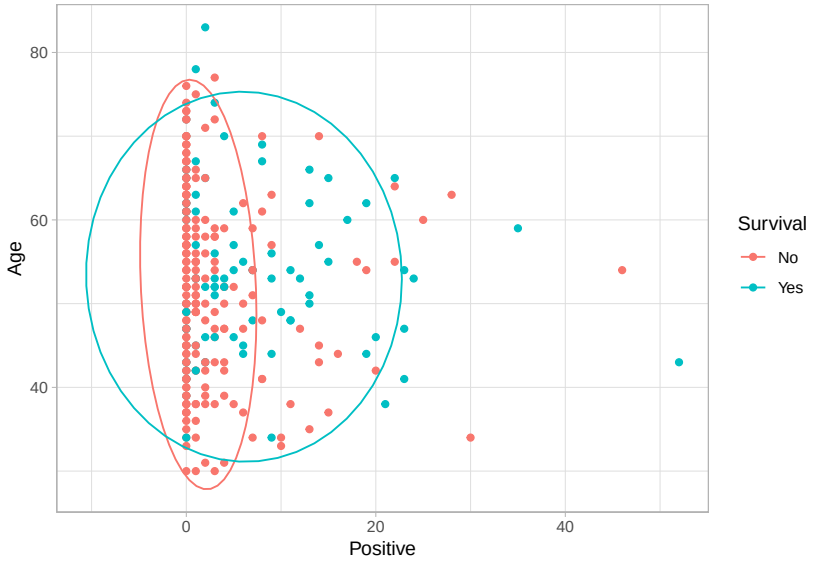

Figura 15

Se nota que la causa de que no se rechace el test para esta variable es la gran cantidad de datos con Positive igual a 0.

Por tanto para LDA no podemos hacer uso de la variable Positive, puesto que además de la falta de normalidad se incumpliría la asunción número 3, por lo que usamos las otras dos.

Aunque solo es recomendable, y no son cualidades necesarias para obtener solución en LDA:

- Tenemos más instancias que predictores, por varios órdenes de magnitud.
- Los predictores son independientes.
- No tenemos varianzas cercanas a cero.

1.2.2. Aplicación del algoritmo LDA

```
Call:
lda(x, y)

Prior probabilities of groups:
       No       Yes
0.7272727 0.2727273

Group means:
            Age        Year
No  -0.05947324 -0.00398263
Yes  0.11192259 -0.03680916

Coefficients of linear discriminants:
            LD1
Age   0.9781149
Year -0.2588776
Cross-Validated (10 fold) Confusion Matrix

(entries are percentual average cell counts across resamples)

          Reference
Prediction   No  Yes
       No  72.7 27.3
       Yes  0.0  0.0

 Accuracy (average) : 0.7273
```

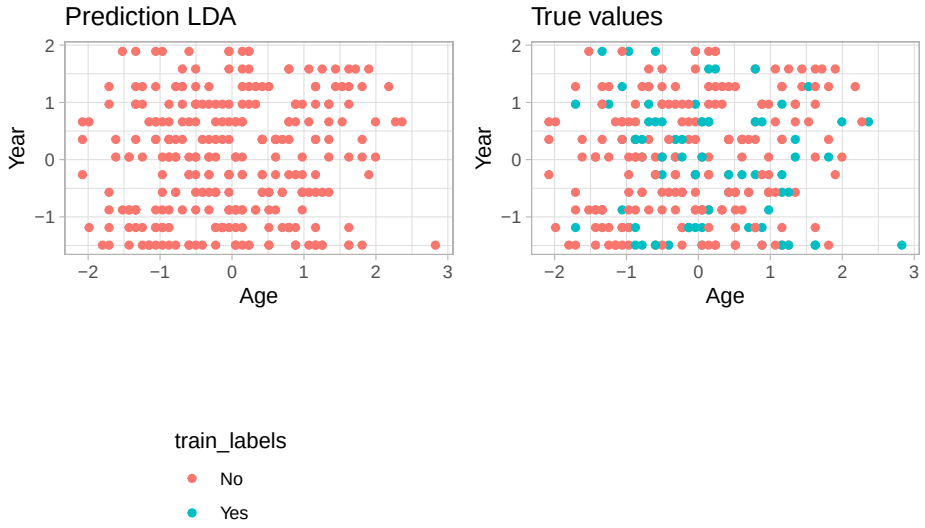

Figura 16: Predicciones LDA sobre training

El gráfico de los discriminantes no muestra una buena separación entre las clases, estando ambos centrados sobre 0.5 y similarmente esparcidos.

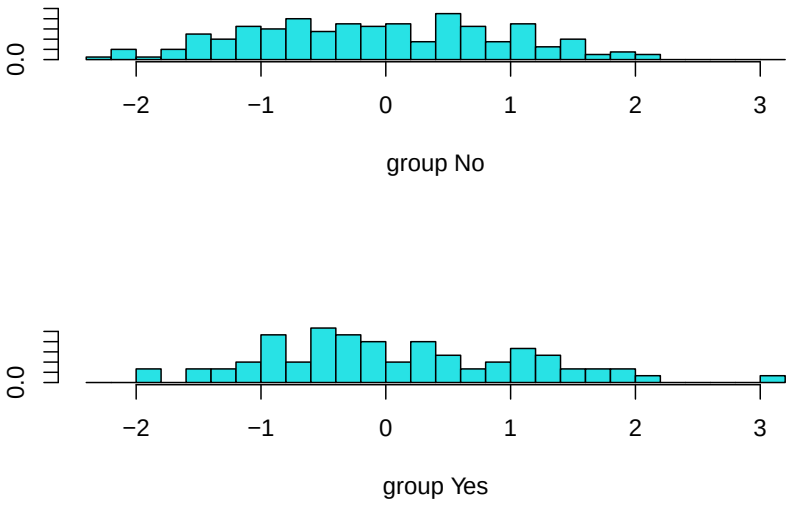

Figura 17: Histograma de los coeficientes de LDA

Y aunque es difícil verlo en 2D, se puede apreciar que el hiperplano que se genera con LDA no separa bien las clases:

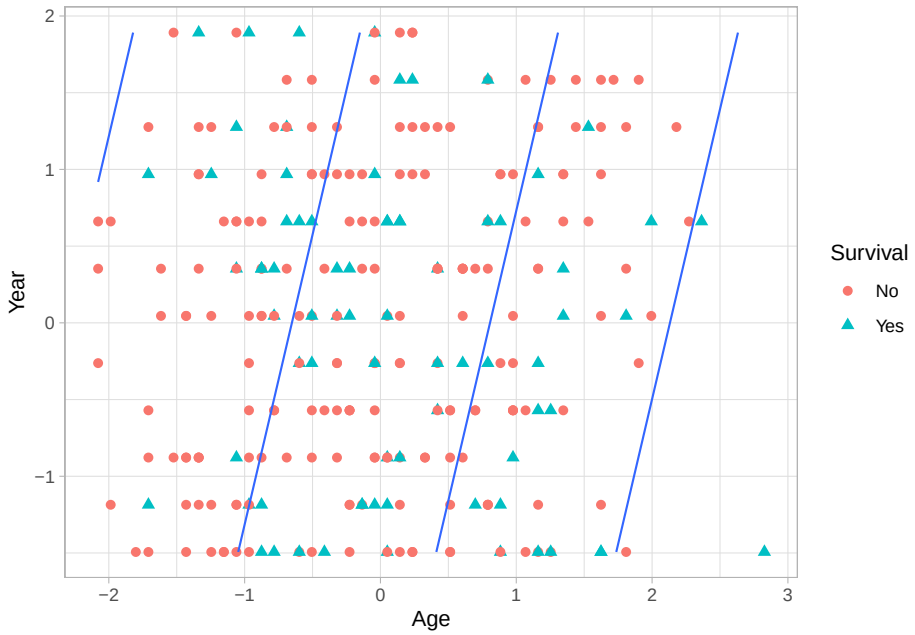

Figura 18
Estamos pintando un contorno 3D y por eso nos salen múltiples líneas en el gráfico

Resultados en test:

```
Test evaluation:
Accuracy     Kappa
0.8064516 0.0000000

Confusion matrix (in test split):
       test_labels
ldaPred No Yes
    No  25   6
    Yes  0   0
```

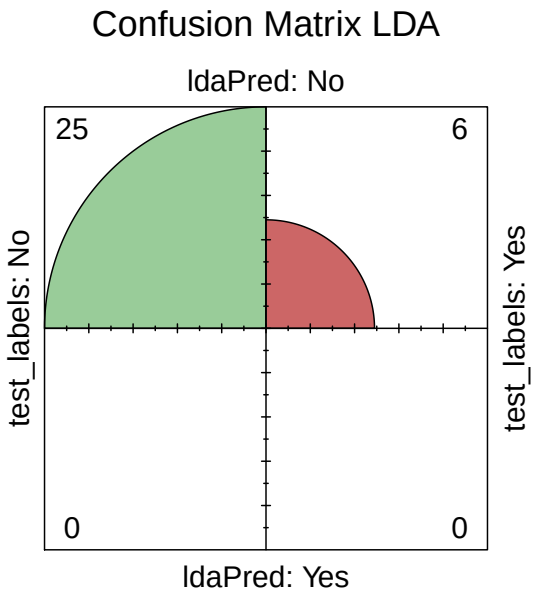

Figura 19: Matriz de confusión sobre el conjunto de test

Tenemos un dataset bastante desbalanceado, y LDA no predice para la clase Yes. Con esto se asegura un alto accuracy en nuestro entrenamiento, pero no asegura de que para datos externos vaya a ser así. Pese a ello, no nos queda más remedio que suponer que nuestros datos vienen de la misma muestra aleatoria y por tanto son releventes para la clasificación.

Adicionalmente los resultados en test nos devuelven un kappa igual a cero, dándonos a entender que los resultados son poco fiables y que se podría obtener esa misma calidad con aleatoriedad pura.

1.3. Algoritmo QDA

1.3.1. Asunciones

QDA tiene las mismas asunciones de LDA salvo que relaja la norma de que las clases tengan igual covarianza. Esto nos permite usar la variable Positive que habíamos descartado en LDA.

Por tanto tenemos los requisitos de:

1. Distribución aleatoria.
2. Distribución normal.

Técnicamente el no cumplir normalidad no imposibilita que se encuentre solución, pero ya no nos lo asegura.

Adicionalmente tenemos de forma recomendada que:

- El número de predictores debe ser menor que el número de instancias de cada clase. Del EDA sabemos que esto es cierto.
- Los predictores dentro de cada clase no deben estar correlacionados. Corroboramos que no se da en las siguientes matrices de correlación.

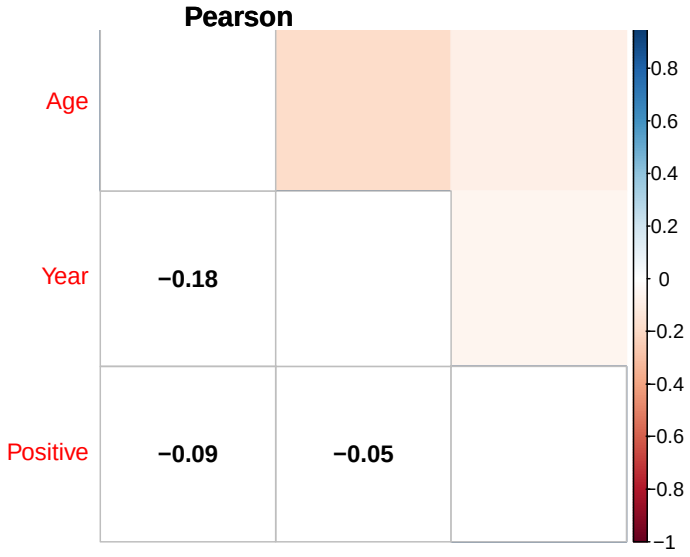

Figura 20: Matriz de correlación para la clase *Yes*

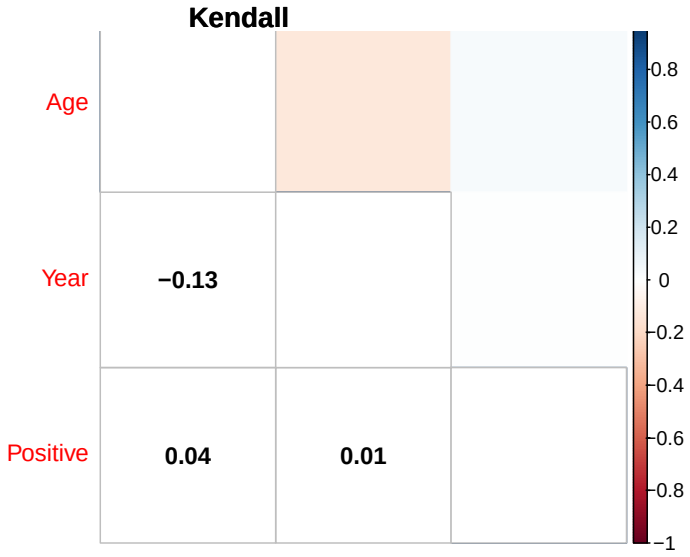

Figura 21: Matriz de correlación para la clase *Yes*

Figura 22: Matriz de correlación para la clase *No*

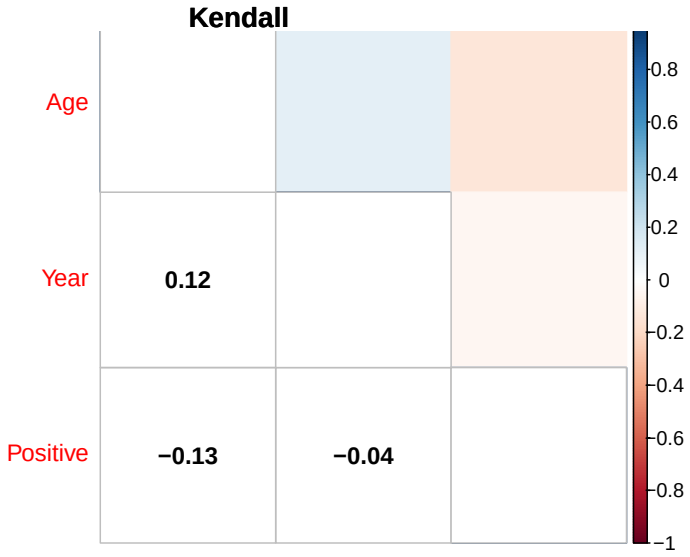

Figura 23: Matriz de correlación para la clase *Yes*

1.3.2. Aplicación del algoritmo QDA

```
Call:
qda(x, y)

Prior probabilities of groups:
       No       Yes
0.7272727 0.2727273

Group means:
            Age        Year   Positive
No  -0.05947324 -0.00398263 -0.1503750
Yes  0.11192259 -0.03680916  0.4933742
Cross-Validated (10 fold) Confusion Matrix

(entries are percentual average cell counts across resamples)

          Reference
Prediction   No  Yes
       No  68.7 21.5
       Yes  4.0  5.8

 Accuracy (average) : 0.7455
```

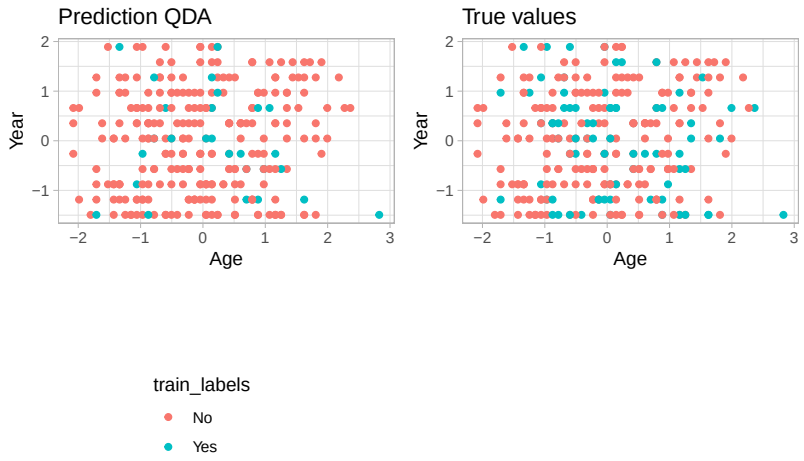

Figura 24: Predicciones QDA sobre training

```
Test evaluation:
Accuracy     Kappa
0.8064516 0.0000000

Confusion matrix (in test split):
       test_labels
qdaPred No Yes
    No  25   6
    Yes  0   0
```

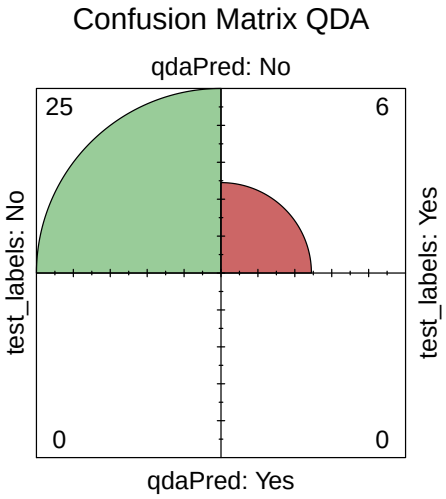

Figura 25: Matriz de confusión sobre el conjunto de test

Obtenemos la mismos accuracy que LDA, pero en este caso vemos que sobre training sí se predice la clase Yes. Por tanto, el hiperplano que forma no será tan restrictivo como el de LDA, pero debería tener una forma similar.

1.4. Comparativa de algoritmos

1.4.1. Para el dataset *haberman*

Si nos fijamos únicamente en los resultados obtenidos para este problema, los tres algoritmos obtienen el mismo accuracy en nuestro conjunto de test[2]. Aunque las etiquetas de este conjunto contienen elementos de ambas clases, podemos ver que se predice mayoritariamente la clase *No*. Como se había mencionado nuestro dataset está bastante desbalanceado, por lo que era más probable que se predijera esa clase con mayor facilidad.

```
Etiquetas:
No  No  Yes No  No  No  Yes No  No  No  No  No  No  No  Yes No  No  Yes Yes
No  No  No  No  No  No  No  Yes No  No  No  No

Predicciones KNN:
No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  No  Yes
No  No  No  No  No  No  No  No  No  No  No  No

Predicciones LDA:
No No No No No No No No No No No No No No No No No No No No No No No No No
No No No No No No

Predicciones QDA:
No No No No No No No No No No No No No No No No No No No No No No No No No
No No No No No No



Accuracy KNN:
 Accuracy     Kappa
0.8387097 0.2439024

Accuracy LDA:
 Accuracy     Kappa
0.8064516 0.0000000

Accuracy QDA:
 Accuracy     Kappa
0.8064516 0.0000000
```

[2]Puesto que hemos usado el paquete caret, no podemos comparar con los datos de accuracy que nos proporciona su salida. Los datos aquí mostrados hacen referencia a una evaluación de los modelos devueltos sobre el conjunto inicialmente reservado de test.

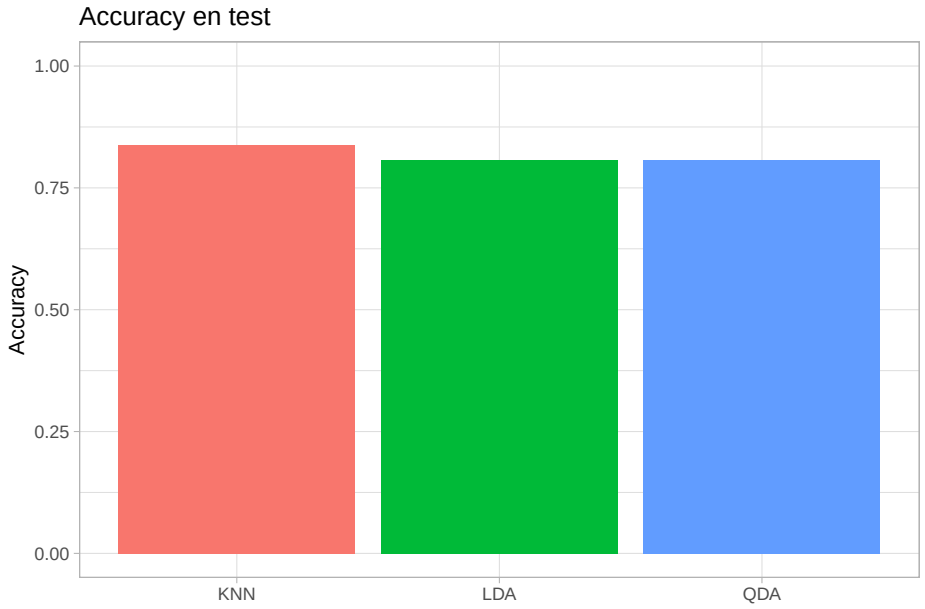

Figura 26

Obtenemos valores de accuracy muy similares, pero diferentes valores de Kappa, siendo en general todos bajos.

Pese a esto, y ya no solo por tener mejores resultados, sino por no cumplir las asunciones necesarias de obtener resultados de calidad en LDA y QDA, para este problema optaríamos por usar el algoritmo KNN.

A partir de las gráficas 3D de las figuras 4 y 5 se apreció una difícil separación de las clases, por lo que el algoritmo de vecinos más cercanos nos resulta una aproximación más lógica.

1.4.2. Comparativas generales

Para comparar la calidad genérica de los algoritmos vamos a aplicar test estadísticos en base a los resultados obtenidos en múltiples datasets.

Estas son las tablas de resultados que tenemos para test:

Dataset	out_test_knn	out_test_lda	out_test_qda
appendicitis	0.8966667	0.8690909	0.8109091
australian	0.6838235	0.8579710	0.8028986
balance	0.9024546	0.8624101	0.9167905
bupa	0.6865775	0.6837924	0.5991759
contraceptive	0.5448653	0.5091561	0.5173102
haberman	0.7462069	0.7481720	0.7512903
hayes-roth	0.5666667	0.5500000	0.5875000
heart	0.6692308	0.8481481	0.8296296
iris	0.9642857	0.9800000	0.9733333
led7digit	0.7510204	0.7420000	0.6975000
mammographic	0.7977698	0.8241269	0.8194042
monk-2	0.9743632	0.7703433	0.9235535
newthyroid	0.9071429	0.9164502	0.9629870
pima	0.7348861	0.7709930	0.7412403
tae	0.3838095	0.5245833	0.5425000
titanic	0.7850353	0.7760304	0.7733032
vehicle	0.6291452	0.7813305	0.8522409
vowel	0.6428571	0.6030303	0.9191919
wine	0.6959559	0.9944444	0.9888889
wisconsin	0.9735023	0.9592185	0.9519476

Aplicamos el test de Wilcoxon a cada pareja de algoritmos:

LDA vs QDA: Obtenemos un ranking de 144 para LDA y 96 para QDA, con un p-valor de 0.75 (o nivel de confianza del 25 %).

```
V = 96, p-value = 0.7562
alternative hypothesis: true location shift is not equal to 0
```

```
  V    V
114 - 96
```

Esto nos dice que LDA obtiene mejores resultados pero puesto que el p-value es extremadamente grande no podemos afirmar con garantía estadística que las diferencias entre los tests sean notorias.

LDA vs KNN: Ahora obtenemos un ranking de 90 para LDA y 120 para QDA, con un p-valor de 0.59 (o nivel de confianza del 41 %).

```
V = 120, p-value = 0.5958
alternative hypothesis: true location shift is not equal to 0
```

```
 V     V
90 - 120
```

Seguimos teniendo un p-valor demasiado grande para poder asegurar la diferencia.

QDA vs KNN: Por último tenemos un ranking de 69 para LDA y 141 para KNN, con un p-valor de 0.18 (o nivel de confianza del 82 %).

```
V = 141, p-value = 0.1893
alternative hypothesis: true location shift is not equal to 0

 V    V
69 - 141
```

Aunque buscaríamos al menos un 95 % de confianza, podemos afirmar al 82 % que los resultados de ambos algoritmos sí son significativamente diferentes.

Una comparativa múltiple de los tres algoritmos con el test de **Friedman** es la siguiente:

```
Friedman rank sum test
Friedman chi-squared = 0.7,
                  df = 2,
             p-value = 0.7047
```

El p-value es mayor que 0.05 por lo que no podemos concluir que haya al menos un par de algoritmos de calidad diferente.

Aunque el resultado del test de Friedman ya nos indica que un análisis post-hoc es innecesario, puesto que los resultados que se obtengan no van a asegurar la diferencia en la calidad de los algoritmos, por completitud en la memoria aplicamos el post-hoc de **Holm**:

```
1 = KNN, 2 = LDA, 3 = QDA

Pairwise comparisons using Wilcoxon signed rank exact test

1    2
2 1.00 -
3 0.53 1.00

P value adjustment method: holm
```

Vemos que los p-value son lo más altos posibles, por lo que carece de sentido intentar diferenciar los algoritmos. Aunque podemos notar, tal y como habíamos visto en los test de Wilcoxon, que la diferencia KNN-QDA probablemente sea mayor que el resto de parejas.

Referencias

[1] http://lib.stat.cmu.edu/datasets/cars.desc.

[2] https://www.ajdesigner.com/phphorsepower/horsepower_equation_trap_speed_method_increase_horsepower.php.

[3] https://statisticsbyjim.com/regression/multicollinearity-in-regression-analysis/#.

[4] http://archive.ics.uci.edu/ml/datasets/Haberman%27s+Survival.

[5] https://www.cancer.org/cancer/breast-cancer/treatment/surgery-for-breast-cancer/lymph-node-surgery-for-breast-cancer.html.

[6] https://en.wikipedia.org/wiki/Lymph_node#.

[7] https://stats.stackexchange.com/questions/82162/cohens-kappa-in-plain-english.

[8] http://thatdatatho.com/2018/02/19/assumption-checking-lda-vs-qda-r-tutorial-2/.