



INTRODUCCIÓN A LA CIENCIA DE DATOS

Trabajo Final Teórico/Práctico

Francisco Pérez Hernández

Tabla de contenido

1 Introducción.....	3
2 Descripción del tipo de datos de entrada	3
2.1 Baseball	3
2.1.1 Cálculo de media, desviación estándar, etc.	4
2.1.2 Gráficos que permitan visualizar los datos adecuadamente.	6
2.1.3 Descripción del conjunto de datos a partir de los puntos anteriores.	8
2.2 Australian	8
2.2.1 Cálculo de media, desviación estándar, etc.	8
2.2.2 Gráficos que permitan visualizar los datos adecuadamente.	9
2.2.3 Descripción del conjunto de datos a partir de los puntos anteriores.	11
3 Regresión	12
3.1 Regresión simple	12
3.2 Regresión múltiple	13
3.3 k-NN para regresión	14
3.4 Comparación de algoritmos	16
4 Clasificación	18
4.1 k-NN	18
4.2 LDA	19
4.3 QDA	19
4.4 Comparación de algoritmos	20

1 Introducción

Las bases de datos con las que he trabajado han sido “baseball” para regresión y “australian” para clasificación:

Dataset Regresión: Baseball

Dataset Clasificación: Australian

Voy a desarrollar lo explicado en clase sobre estos dataset y a analizar los resultados obtenidos. En la primera sección se analizarán las bases de datos asignadas donde se conocerán las estructuras y cuál es el objetivo de cada una de ellas. Esta información se ha extraído de la web Keel, junto a sus archivos asociados, los cuales los he conseguido de los siguientes enlaces:

Baseball: <http://sci2s.ugr.es/keel/dataset.php?cod=76#sub1>

Australian: <http://sci2s.ugr.es/keel/dataset.php?cod=53#sub1>

De forma que se ha podido extender la información facilitada en los documentos subidos por los profesores.

En la segunda sección se realiza un estudio de regresión sobre el dataset “baseball”, donde buscaré modelos que expliquen el problema. Compararé estos modelos y sus ajustes para encontrar el mejor. Finalmente realizaré el test de Wilcoxon y Friedman para los distintos modelos.

En la tercera sección realizaré un estudio similar a regresión, pero esta vez será sobre el dataset “australian” y es un problema de clasificación.

2 Descripción del tipo de datos de entrada

2.1 Baseball

El dataset Baseball contiene 17 variables/columnas de las que 16 son entradas y 1 salida. Las entradas son: “Batting_average”, “On-base_percentage”, “Runs”, “Hits”, “Doubles”, “Triples”, “HomeRuns”, “Runs_batted_in”, “Walks”, “Strike-Outs”, “Stolen_bases”, “Errors”, “Free_agency_eligibility”, “Free_agent”, “Arbitration_eligibility”, “Arbitration”. La salida es “Salary”.

Tenemos en total 337 instancias/observaciones y en ninguna hay valores perdidos. Son datos reales, y los tipos de cada variable son 2 reales y 14 enteros. Las dos variables reales son “Batting_average” y “On-base_percentage”.

Como podemos ver en la web, tenemos que, de las 14 variables de números enteros, tenemos 4 variables que solo contiene valores de 0 o 1: “Free_agency_eligibility”, “Free_agent”, “Arbitration_eligibility” y “Arbitration”.

Como información adicional tenemos que: “El dataset contiene los salarios de 1992 de un conjunto de jugadores de la “Major League Baseball” que jugaron al menos un partido en la temporada 1991 y 1992, excluyendo a los pitchers (lanzadores). Para cada jugador, se proporcionan algunas medidas de rendimiento junto con cuatro variables categóricas que indican cuán libre era cada uno para formar parte de otros equipos.”

2.1.1 Cálculo de media, desviación estándar, etc.

Para obtener la información la distribución de los datos, su media, valores mínimos, máximos, cuartiles... se ha hecho la función `summary()`:

```
> summary(baseball)
   Batting_average    On-base_percentage      Runs       Hits       Doubles      Triples
Min.   :0.0630      Min.   :0.063            Min.   : 0.0  Min.   : 1.00  Min.   : 0.00  Min.   : 0.000
1st Qu.:0.2380     1st Qu.:0.297          1st Qu.: 22.0 1st Qu.: 51.00 1st Qu.: 9.00  1st Qu.: 0.000
Median :0.2600      Median :0.323          Median : 41.0 Median : 91.00 Median :15.00  Median : 2.000
Mean   :0.2578      Mean   :0.324          Mean   : 46.7  Mean   : 92.83  Mean   :16.67  Mean   : 2.338
3rd Qu.:0.2810     3rd Qu.:0.354          3rd Qu.: 69.0 3rd Qu.:136.00 3rd Qu.:23.00  3rd Qu.: 3.000
Max.   :0.4570      Max.   :0.486          Max.   :133.0  Max.   :216.00  Max.   :49.00  Max.   :15.000
   HomeRuns      Runs_batted_in      Walks      Strike-Outs      Stolen_bases      Errors
Min.   : 0.000      Min.   : 0.00           Min.   : 0.00  Min.   : 1.00  Min.   : 0.0000  Min.   : 0.000
1st Qu.: 2.000     1st Qu.: 21.00        1st Qu.: 15.00 1st Qu.: 31.00 1st Qu.: 1.000  1st Qu.: 3.000
Median : 6.000      Median : 39.00        Median : 30.00  Median : 49.00  Median : 4.000  Median : 5.000
Mean   : 9.098      Mean   : 44.02        Mean   : 35.02  Mean   : 56.71  Mean   : 8.246  Mean   : 6.772
3rd Qu.:15.000     3rd Qu.: 66.00        3rd Qu.: 49.00 3rd Qu.: 78.00 3rd Qu.:11.000 3rd Qu.: 9.000
Max.   :44.000      Max.   :133.00        Max.   :138.00  Max.   :175.00  Max.   :76.000  Max.   :31.000
   Free_agency_eligibility  Free_agent      Arbitration_eligibility  Arbitration      Salary
Min.   :0.0000      Min.   :0.0000        Min.   :0.00000      Min.   :0.00000  Min.   : 109
1st Qu.:0.0000     1st Qu.:0.0000        1st Qu.:0.00000     1st Qu.:0.00000  1st Qu.: 230
Median :0.0000      Median :0.0000        Median :0.00000      Median :0.00000  Median : 740
Mean   :0.3976      Mean   :0.1157        Mean   :0.1929       Mean   :0.02967  Mean   :1249
3rd Qu.:1.0000     3rd Qu.:0.0000        3rd Qu.:0.00000     3rd Qu.:0.00000  3rd Qu.:2150
Max.   :1.0000      Max.   :1.0000        Max.   :1.00000      Max.   :1.00000  Max.   :6100
```

Figura 1: Resumen de distribución de los datos en cada variable para el dataset “baseball”

Si queremos la desviación estándar, con `apply()` podemos obtener la información:

```
> desviacion_estandar_baseball
   Batting_average    On-base_percentage      Runs       Hits
 3.954634e-02      4.713206e-02      2.902017e+01  5.189632e+01
   Doubles           Triples           HomeRuns      Runs_batted_in
 1.045200e+01      2.543336e+00      9.289934e+00  2.955941e+01
   Walks            Strike-Outs      Stolen_bases      Errors
 2.484247e+01      3.382878e+01      1.166478e+01  5.927490e+00
   Free_agency_eligibility  Free_agent  Arbitration_eligibility  Arbitration
 4.901351e-01      3.203730e-01      3.951450e-01  1.699375e-01
   Salary
 1.240013e+03
```

Figura 2: Desviación estándar para el dataset “baseball”

Para la varianza:

```
> varianza_baseball
   Batting_average    On-base_percentage      Runs       Hits
 1.563913e-03      2.221431e-03      8.421700e+02  2.693228e+03
   Doubles           Triples           HomeRuns      Runs_batted_in
 1.092443e+02      6.468560e+00      8.630288e+01  8.737585e+02
   Walks            Strike-Outs      Stolen_bases      Errors
 6.171485e+02      1.144387e+03      1.360671e+02  3.513514e+01
   Free_agency_eligibility  Free_agent  Arbitration_eligibility  Arbitration
 2.402324e-01      1.026388e-01      1.561396e-01  2.887876e-02
   Salary
 1.537633e+06
```

Figura 3: Varianza para el dataset “baseball”

Para la desviación absoluta de la mediana:

	Batting_average	On-base_percentage	Runs	Hits
	0.0311346	0.0429954	34.0998000	62.2692000
	Doubles	Triples	HomeRuns	Runs_batted_in
	10.3782000	1.4826000	7.4130000	29.6520000
	Walks	Strike-Outs	Stolen_bases	Errors
	23.7216000	34.0998000	5.9304000	4.4478000
Free_agency_eligibility	Free_agent	Arbitration_eligibility		Arbitration
0.0000000	0.0000000	0.0000000		0.0000000
Salary				
879.1818000				

Figura 4: Desviación absoluta de la mediana para el dataset “baseball”

Para el rango intercuartil:

	Batting_average	On-base_percentage	Runs	Hits
	0.043	0.057	47.000	85.000
	Doubles	Triples	HomeRuns	Runs_batted_in
	14.000	3.000	13.000	45.000
	Walks	Strike-Outs	Stolen_bases	Errors
	34.000	47.000	10.000	6.000
Free_agency_eligibility	Free_agent	Arbitration_eligibility		Arbitration
1.000	0.000	0.000		0.000
Salary				
1920.000				

Figura 5: Rango intercuartil para el dataset “baseball”

2.1.2 Gráficos que permitan visualizar los datos adecuadamente.

Si queremos ver las gráficas de todas las variables con respecto a todas, podemos hacer uso de la función `plot()`

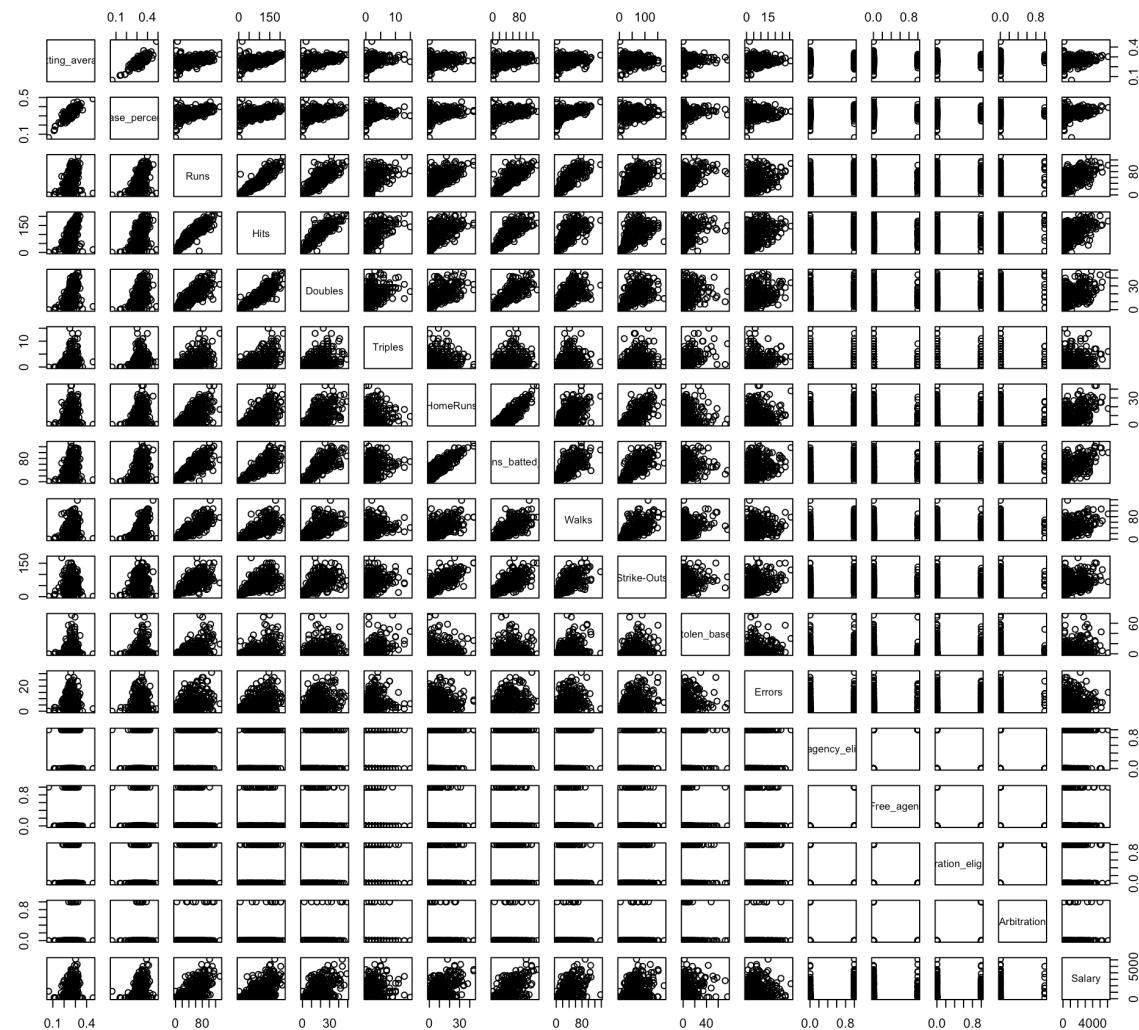


Figura 6: Plot de todas las variables del dataset "baseball"

Pero si lo que queremos es ver todas las variables con respecto de la variable de salida, es decir *Salary*, entonces tendremos:

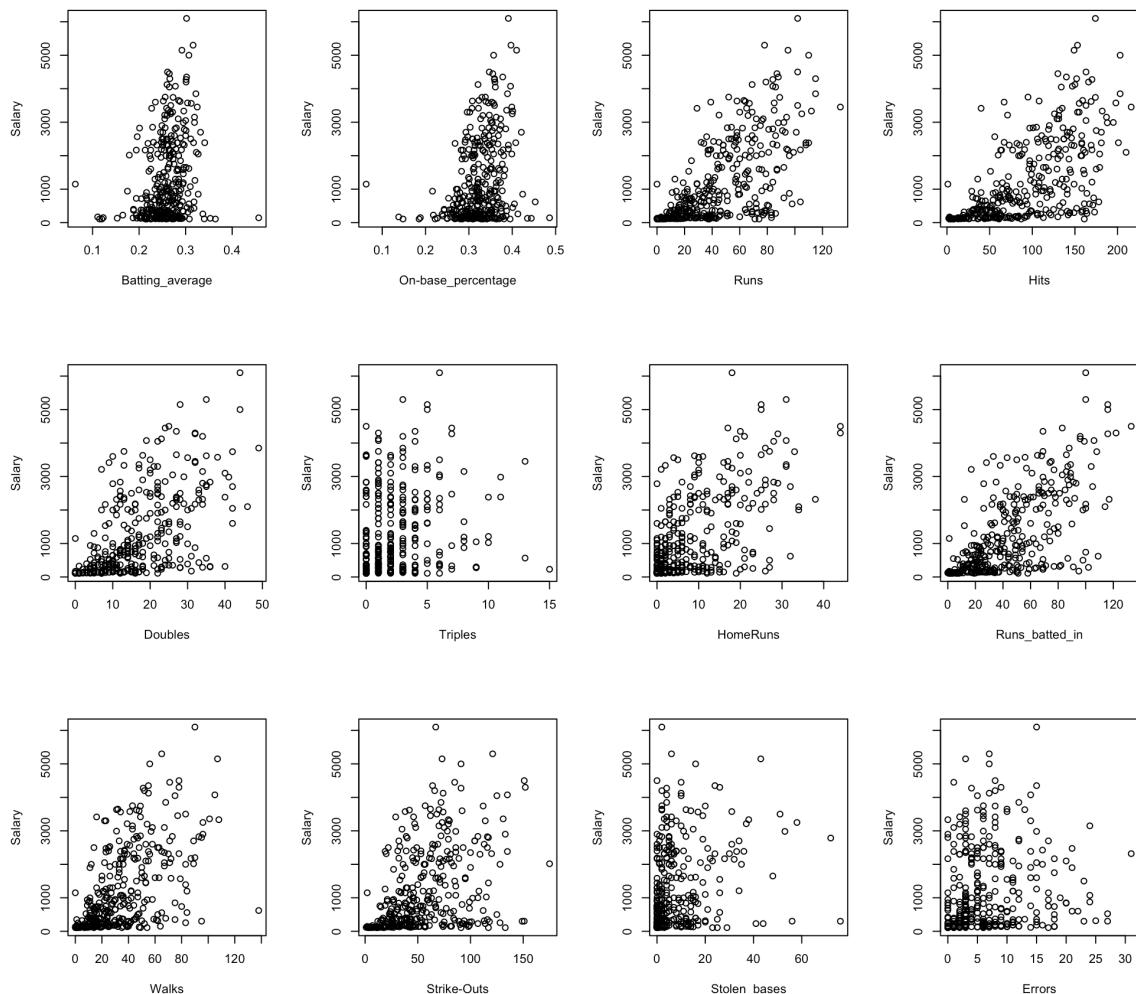


Figura 7: Plot de las primeras 12 variables

Además, tendremos:

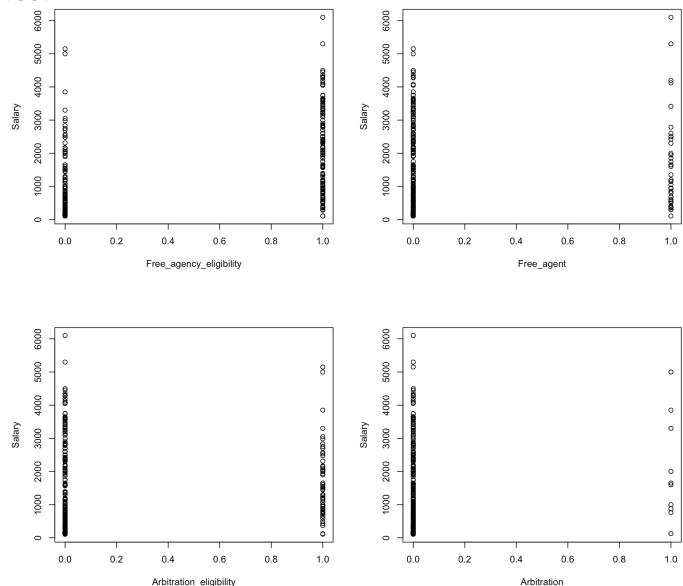


Figura 8: Plot de las variables con valores de 0 o 1.

2.1.3 Descripción del conjunto de datos a partir de los puntos anteriores.

En resumen, tenemos un problema de regresión en el que se pretende realizar un modelo para obtener como variable respuesta el salario de los jugadores de un equipo de béisbol. Para ello contamos con 16 variables, de las cuales 4 se mueven entre valores de 0 o 1 y el resto son enteros o valores reales. Hemos podido observar la distribución de estas variables y sus gráficas con respecto a la variable de salida. A simple vista no se podría decir mucho sobre qué variables son las más importantes, pero puede decirse que algunas de las más influyentes pueden ser: "Hits", "Runs", "HomeRuns" ... por lo cual miraremos la matriz de correlación.

Como la correlación simple se da con valores cercanos a 1 o -1, indicando alta correlación entre cada par de variables, y valores cercanos a 0 indican el caso contrario, se tiene que las variables correladas son:

	Runs	Hits	Doubles	HomeRuns
Runs_batted_in	0.6429035	0.6212390	0.5774227	0.5904536
Walks	0.6684219	0.5670848	0.5645857	1.0000000
Free_agency_eligibility				Salary

Figura 9: Correlación simple entre cada variable con la variable respuesta

Donde tenemos "Runs", "Hits", "Doubles", "HomeRuns", "Runs_batted_in", "Walks" y "Free_agency_eligibility". Es decir, estas son las variables que tendremos más en cuenta para nuestros modelos.

2.2 Australian

El dataset Australian contiene 15 variables/columnas de las que 14 son entradas y 1 salida. Las variables de este dataset no tienen ningún nombre por lo que no podemos describirlas. Este es un problema de clasificación en el que tenemos 2 clases, la 0 y la 1.

Tenemos en total 690 instancias/observaciones y en ninguna hay valores perdidos. Son datos reales, y los tipos de cada variable son 3 reales, 5 enteros y 6 nominales.

Como podemos ver en la web, tenemos que, de las 14 variables de números enteros, tenemos 4 que se mueven entre 0 o 1. Estas son las variables 1, 8, 9 y 11, además de la de salida, la 15.

Como información adicional tenemos que: "El archivo concierne a la aplicación de tarjetas de crédito. Todos los nombres de los atributos y los valores han sido cambiados para proteger la confidencialidad de los datos. El dataset es interesante porque hay una buena mezcla de atributos: continuos, nominales con pequeños valores de números, y nominales con grandes valores numéricos."

2.2.1 Cálculo de media, desviación estándar, etc.

Para obtener la información la distribución de los datos, su media, valores mínimos, máximos, cuartiles... se ha hecho la función `summary()`:

```
> summary(australian)
      V1          V2          V3          V4          V5          V6
Min. :0.0000  Min. : 16  Min. :  0  Min. :1.000  Min. : 1.000  Min. :1.000
1st Qu.:0.0000  1st Qu.:1942  1st Qu.: 15  1st Qu.:2.000  1st Qu.: 4.000  1st Qu.:4.000
Median :1.0000  Median :2629  Median : 125  Median :2.000  Median : 8.000  Median :4.000
Mean   :0.6783  Mean   :2697  Mean   :1187  Mean   :1.767  Mean   : 7.372  Mean   :4.693
3rd Qu.:1.0000  3rd Qu.:3525  3rd Qu.: 665  3rd Qu.:2.000  3rd Qu.:10.000  3rd Qu.:5.000
Max.  :1.0000  Max.  :8025  Max.  :26335  Max.  :3.000  Max.  :14.000  Max.  :9.000
      V7          V8          V9          V10         V11         V12
Min. : 0.0  Min. :0.0000  Min. :0.0000  Min. : 0.0  Min. :0.0000  Min. :1.000
1st Qu.: 5.0  1st Qu.:0.0000  1st Qu.:0.0000  1st Qu.: 0.0  1st Qu.:0.000  1st Qu.:2.000
Median : 35.0  Median :1.0000  Median :0.0000  Median : 0.0  Median :0.000  Median :2.000
Mean   :453.4  Mean   :0.5232  Mean   :0.4275  Mean   : 2.4  Mean   :0.458  Mean   :1.929
3rd Qu.:219.8  3rd Qu.:1.0000  3rd Qu.:1.0000  3rd Qu.: 3.0  3rd Qu.:1.000  3rd Qu.:2.000
Max. :14415.0  Max. :1.0000  Max. :1.0000  Max. :67.0  Max. :1.000  Max. :3.000
      V13         V14         V15
Min. : 0  Min. : 1.0  Min. :0.0000
1st Qu.: 80 1st Qu.: 1.0  1st Qu.:0.0000
Median :160  Median : 6.0  Median :0.0000
Mean   :184  Mean   :1018.4  Mean   :0.4449
3rd Qu.:272  3rd Qu.: 396.5 3rd Qu.:1.0000
Max. :2000  Max. :100001.0  Max. :1.0000
```

Figura 10: Resumen de distribución de los datos en cada variable para el dataset “australian”

Si queremos la desviación estándar, con *apply()* podemos obtener la información:

```
> desviacion_estandar_australian
      V1          V2          V3          V4          V5          V6          V7          V8
0.4674824 1554.5597320 3069.1100423 0.4300628 3.6832648 1.9923161 1387.9003240 0.4998243
      V9          V10         V11         V12         V13         V14         V15
0.4950800 4.8629400 0.4985919 0.2988131 172.1592735 5210.1025983 0.4973183
```

Figura 11: Desviación estándar para el dataset “australian”

Para la varianza:

```
> varianza_australian
      V1          V2          V3          V4          V5          V6          V7          V8
2.185398e-01 2.416656e+06 9.419436e+06 1.849540e-01 1.356644e+01 3.969323e+00 1.926267e+06 2.498244e-01
      V9          V10         V11         V12         V13         V14         V15
2.451042e-01 2.364819e+01 2.485938e-01 8.928925e-02 2.963882e+04 2.714517e+07 2.473255e-01
```

Figura 12: Varianza para el dataset “australian”

Para la desviación absoluta de la mediana:

```
> desviacion_absoluta_de_la_mediana_australian
      V1          V2          V3          V4          V5          V6          V7          V8          V9          V10         V11
0.0000 1166.8062 177.9120 0.0000 4.4478 0.0000 51.8910 0.0000 0.0000 0.0000 0.0000
      V12         V13         V14         V15
0.0000 148.2600 7.4130 0.0000
```

Figura 13: Desviación absoluta de la mediana para el dataset “australian”

Para el rango intercuartil:

```
> rango_intercuartil_australian
      V1          V2          V3          V4          V5          V6          V7          V8          V9          V10         V11         V12         V13
1.00 1583.00 650.00 0.00 6.00 1.00 214.75 1.00 1.00 3.00 1.00 0.00 192.00
      V14         V15
395.50 1.00
```

Figura 14: Rango intercuartil para el dataset “australian”

2.2.2 Gráficos que permitan visualizar los datos adecuadamente.

Si queremos ver las gráficas de todas las variables con respecto a todas, podemos hacer uso de la función *plot()*

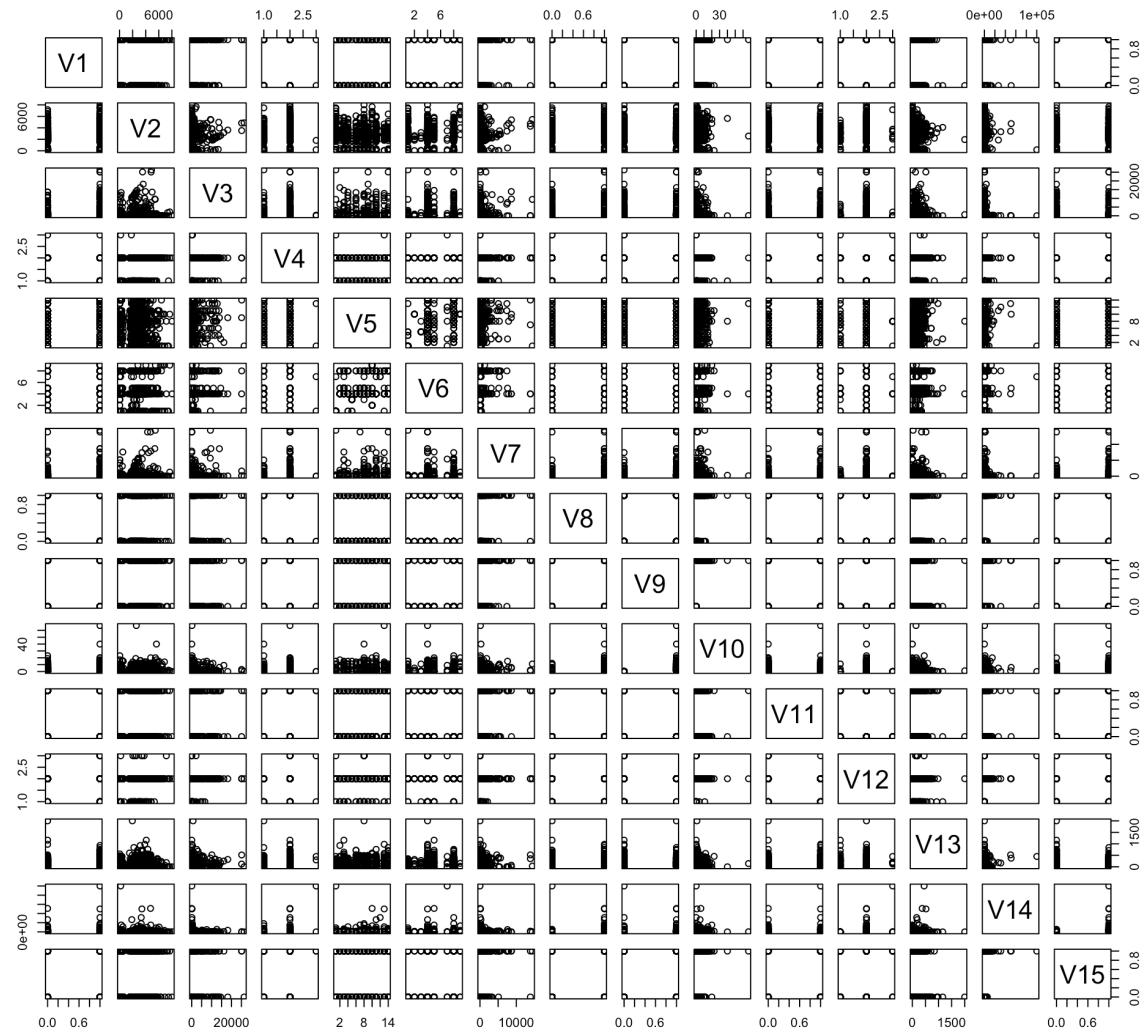


Figura 15: Plot de todas las variables del dataset “australian”

Pero si lo que queremos es ver todas las variables con respecto de la variable de salida, es decir la variable 15, entonces tendremos:

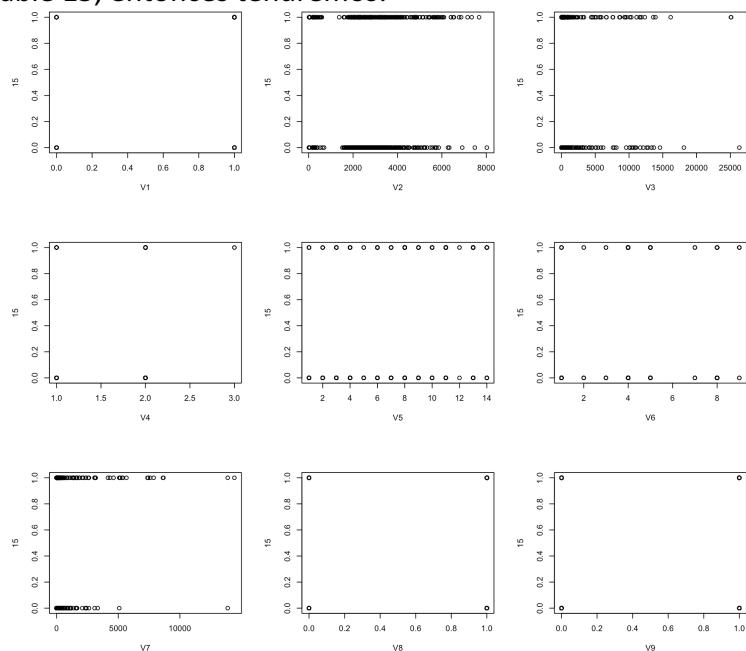


Figura 16: Plot de las primeras 9 variables

Además, tendremos:

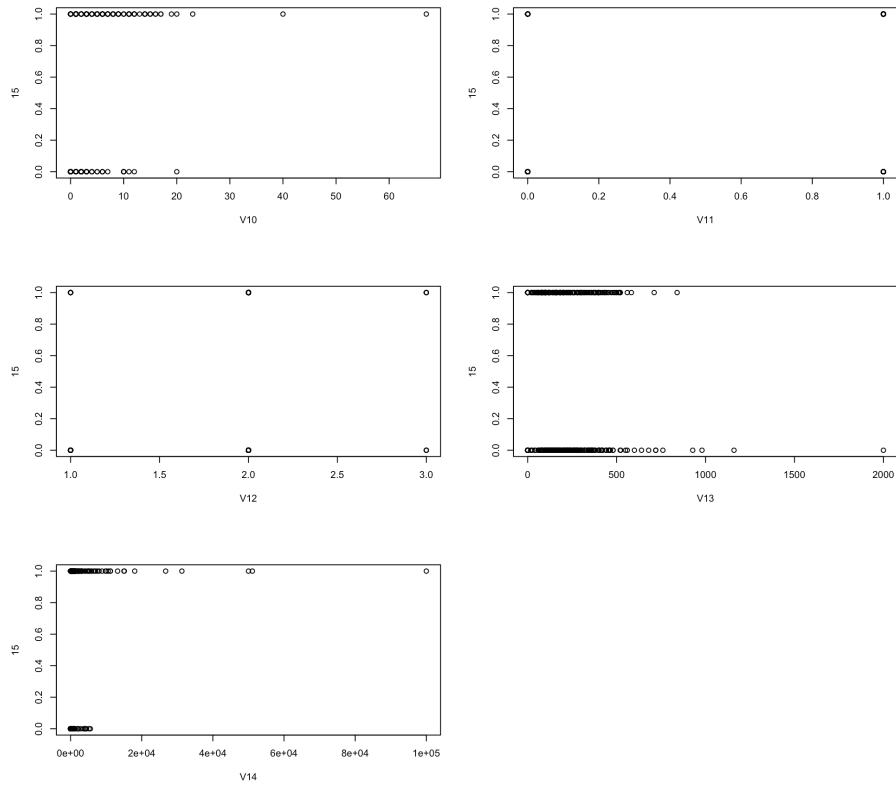


Figura 17: Plot del resto de variables.

2.2.3 Descripción del conjunto de datos a partir de los puntos anteriores.

En resumen, tenemos un problema de clasificación en el que se pretende realizar un modelo para obtener como variable respuesta una de las dos clases. Para ello contamos con 14 variables, de las cuales 4 se mueven entre valores de 0 o 1 y el resto son enteros, valores reales o nominales. Hemos podido observar la distribución de estas variables y sus gráficas con respecto a la variable de salida.

```
> correlacion_australian[cor(australian)[,15] > 0.5 | cor(australian)[,15] < -0.5]
      V8          V15
0.7204068 1.0000000
```

Figura 18: Correlación simple entre cada variable con la variable respuesta.

A simple vista no se podría decir mucho sobre que variables son las más importantes, pero con ayuda de la matriz de correlación, vemos como la Variable 8 está fuertemente correlada con la variable predictora, por lo que la tendremos muy en cuenta cuando realicemos nuestros modelos.

3 Regresión

Para la base de datos “baseball” voy a aplicar regresión, de forma que vamos a obtener diferentes modelos, tanto para regresión simple, regresión múltiple y para knn, de forma que finalmente los compararemos para ver cuál se adapta mejor al problema.

3.1 Regresión simple

Para el estudio de la regresión simple, voy a usar las variables que tienen más correlación ya que nuestro dataset tiene más de 5 variables. Estas variables nos habían salido:

Runs	Hits	Doubles	HomeRuns
0.6429035	0.6212390	0.5774227	0.5904536
Runs_batted_in	Walks	Free_agency_eligibility	Salary
0.6684219	0.5670848	0.5645857	1.0000000

Figura 19: Correlación simple entre cada variable con la variable respuesta

Donde tenemos “Runs”, “Hits”, “Doubles”, “HomeRuns”, “Runs_batted_in”, “Walks” y “Free_agency_eligibility”. Por lo tanto, yo voy a elegir como las más importantes a: “Runs”, “Hits”, “Runs_batted_in”, “HomeRuns” y “Doubles”. Veamos que tal salen estos modelos de regresión simple. La visualización de estos modelos nos queda:

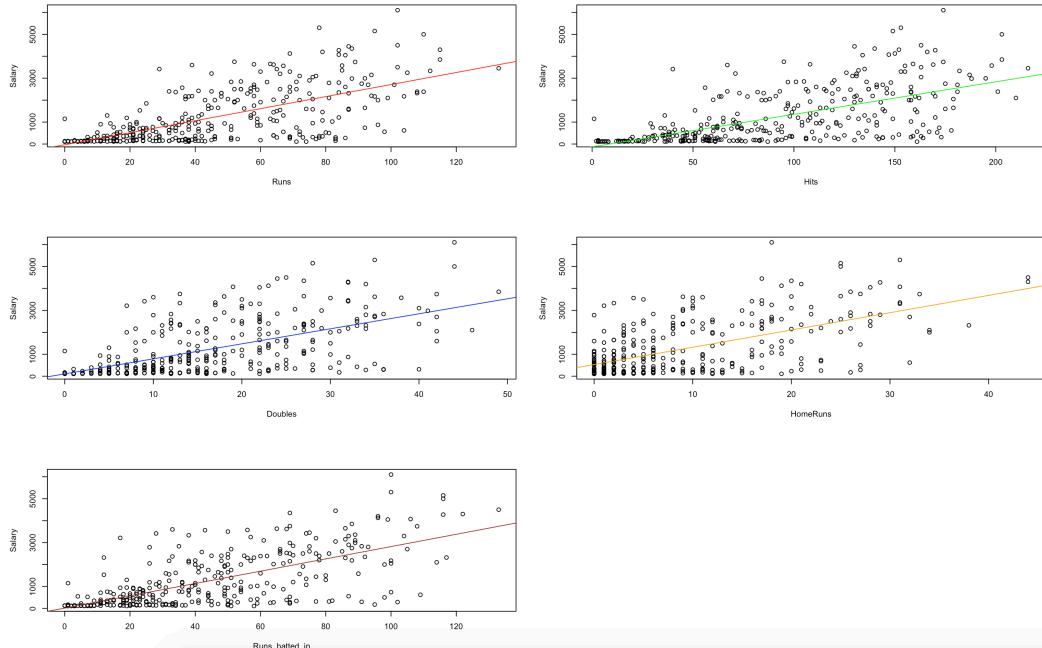


Figura 20: Visualización de los modelos

Además, si queremos ver los datos más importantes:

	adj.r.squared	p-value	RMSE
fitRuns	0.4115736	1.085793e-40	951.2013
fitHits	0.3841049	2.340826e-37	973.1499
fitDoubles	0.3314271	2.347741e-31	1013.9131
fitHomeRuns	0.3466911	4.798143e-33	1002.2721
fitRunsBattedIn	0.4451365	5.582361e-45	923.6755

Figura 21: Datos más importantes de los modelos

De donde vemos como los modelos tiene un bajo valor de R^2 por lo que debemos seguir mejorando los modelos.

Como el siguiente paso a realizar es regresión múltiple, y en este curso hemos estudiado regresión mediante cross-validation, he realizado las funciones que se dieron en la sesión de laboratorio, pero modificadas para obtener los datos más interesantes. De esta forma, tras aplicarlo sobre los modelos arriba vistos, de regresión simple, comparando además un modelo de regresión múltiple con todas las variables, obtenemos lo siguiente:

	EER	T1	R2	T2	R2_Adj	T3	MSEtrain	MSEtest	RMSE
model1	76	No	0.417	No	0.415	Si	894529	914191.4	949.2
model2	77.8	No	0.39	No	0.388	Si	935782.9	948554.4	970.8
model3	81.2	No	0.336	No	0.333	Si	1020714	1039593	1013.8
model4	80.4	No	0.349	No	0.347	Si	998913.7	1014911	1003.2
model5	74	No	0.448	No	0.446	Si	848142.2	858750.2	924.2
model6	55.4	No	0.708	No	0.689	No	448159.2	536675.5	672

Figura 22: Datos más importantes de los modelos

Por lo que vemos que el modelo con todas las variables (modelo6) obtiene mejores valores que el resto de modelos de regresión simple, con un 68,9% de R2 ajustado.

3.2 Regresión múltiple

Como se ha dicho antes, para realizar regresión múltiple se han usado las funciones vistas en el curso para hacer cross-validation. Además, en el código entregado, yo he realizado los modelos sin cross-validation, para observar que se obtienen unos valores casi iguales.

Lo primero que se ha realizado han sido modelos donde se han usado todas las variables y a partir de este se han ido eliminando las que parecían menos importantes o que no ayudaban a explicar el modelo.

Seguidamente se han intentado hacer interacciones y aplicar no linealidad a los modelos, de forma que hemos ido progresando en los resultados.

Finalmente hemos realizado 21 modelos más los 5 modelos simples.

Los resultados de estos modelos son los que se muestran en la figura 23:

	EER	T1	R2	T2	R2_Adj	T3	MSEtrain	MSEtest	RMSE
model1	76	No	0.417	No	0.415	Si	894529	914191.4	949.2
model2	77.8	No	0.39	No	0.388	Si	935782.9	948554.4	970.8
model3	81.2	No	0.336	No	0.333	Si	1020714	1039593	1013.8
model4	80.4	No	0.349	No	0.347	Si	998913.7	1014911	1003.2
model5	74	No	0.448	No	0.446	Si	848142.2	858750.2	924.2
model6	55.4	No	0.708	No	0.689	No	448159.2	536675.5	672
model7	55.4	No	0.705	No	0.69	No	452255.6	514877.4	675
model8	55.4	No	0.703	No	0.69	No	455805.1	514594.1	677.6
model9	55.4	No	0.702	No	0.69	No	457401.8	514800.6	678.8
model10	55.4	No	0.7	No	0.69	No	459832.4	513605.6	680.6
model11	55.7	No	0.695	No	0.687	Si	467885.9	504836.1	686.6
model12	44.3	No	0.848	Si	0.801	No	232700.9	1453703	484.2
model13	51.9	No	0.775	No	0.727	No	344910.7	1495766	589.2
model14	44.3	No	0.848	Si	0.801	No	232700.9	1453703	484.2
model15	44.3	No	0.848	Si	0.801	No	232700.9	1453703	484.2
model16	44.3	No	0.849	Si	0.802	No	232270	20502528	483.6
model17	44.5	No	0.847	Si	0.8	No	234926.9	75755213	486.2
model18	55.1	No	0.701	No	0.693	Si	458501.5	507286.5	679.6
model19	82.5	No	0.322	No	0.312	No	1040491	1124631	1023.6
model20	50.5	No	0.747	No	0.742	Si	388618	412666.4	625.4
model21	50.5	No	0.745	No	0.742	Si	390324.4	413702.4	626.6
model22	51.3	No	0.737	No	0.734	Si	403834.6	415978.1	637.6
model23	49.4	No	0.762	No	0.753	Si	364055.2	429010.1	605.4
model24	48.5	No	0.787	No	0.762	No	327314.3	487122	574
model25	46.8	No	0.817	Si	0.778	No	280797.3	541790.8	531.6
model26	45.8	No	0.831	Si	0.788	No	258709.3	6896936	510

Figura 23: Datos más importantes de los modelos

El modelo que ha obtenido un menor MSE en test ha sido el modelo 20 con un R2 y R2adj del 0.742, explicando un 74% del problema. El modelo que mejor MSE en train obtiene es el modelo 16 con muy buen R2. El modelo con mayor MSE en train es el 19 y el que mayor MSE en test obtiene es el 17, indicando sobre-ajuste del modelo. El modelo 16 es el que obtiene un mayor valor de R2 y R2adj.

```
> df1m[c(16,20),]
   EER T1      R2 R2_Adj T3 MSEtrain MSEtest RMSE
model16 44.3 No 0.849 Si  0.802 No  232270 20502528 483.6
model20 50.5 No 0.747 No  0.742 Si   388618 412666.4 625.4
```

Figura 24: Datos más importantes de los modelos

Por lo tanto, de los mejores modelos obtenidos tenemos al modelo 16 y el modelo 20. Finalmente me decantaré a elegir como el mejor al modelo 20 al tener un menor valor de MSE en test.

Este modelo es el siguiente:

```
> summary(fitM20)
```

```
Call:
lm(formula = Salary ~ I(Runs_batted_in^2) * Free_agency_eligibility +
(`Strike-Outs` + Stolen_bases) * Free_agency_eligibility +
I((Runs_batted_in + Stolen_bases)^3) * Arbitration_eligibility +
(Batting_average + `On-base_percentage` + Runs + Hits + Doubles +
Triples + HomeRuns + Walks + Errors) * Free_agency_eligibility +
(Batting_average + `On-base_percentage` + Runs + Hits + Doubles +
Triples + HomeRuns + Walks + Errors) * Arbitration_eligibility +
(Batting_average + `On-base_percentage` + Runs + Hits + Doubles +
Triples + HomeRuns + Walks + Errors) * Free_agent, data = baseball)
```

Figura 25: modelo20

O también:

```
> modelo20
Y ~ I(X8 * X13^2) + I(X8 * X15^2) + I(X11 * X13^2) + I(X11 *
X15^2) + I(X7 * X10 * X14^2)
```

Figura 26: modelo20

El modelo se ve que es bastante difícil de explicar, pero nos explica un 74% del problema.

3.3 k-NN para regresión

A continuación, pasamos a knn para regresión. Lo primero que he realizado ha sido como se mostraba en las trasparencias del laboratorio, un modelo knn con todos los datos del dataset para modelo simples. Estos modelos visualizados nos dan el siguiente resultado:

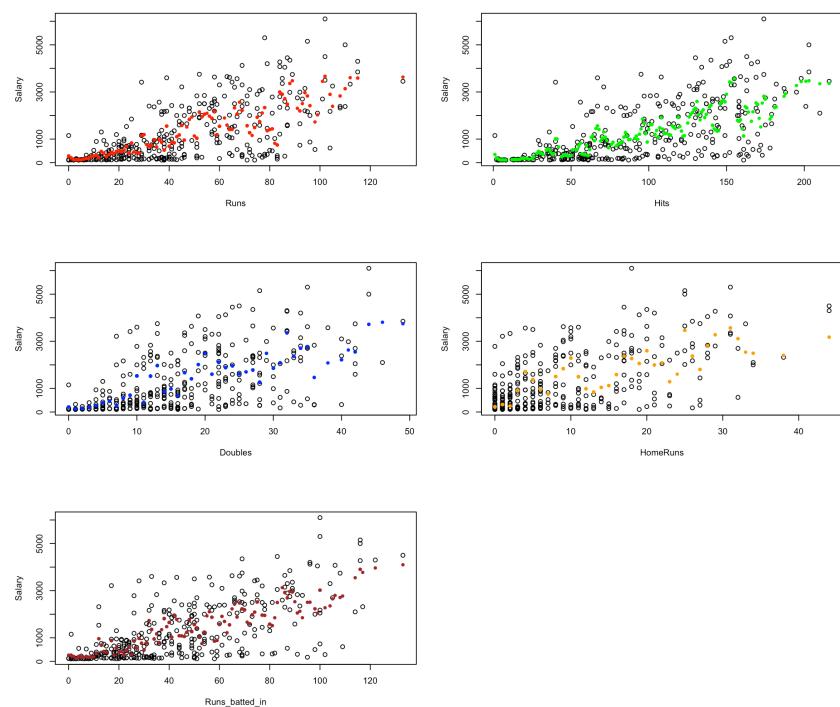


Figura 27: visualización de los modelos

```
> regresion_simple_knn
      adj.r.squared      p-value      RMSE   RMSEknn
fitRuns          0.4115736 1.085793e-40 951.2013 838.0471
fitHits          0.3841049 2.340826e-37 973.1499 859.0143
fitDoubles       0.3314271 2.347741e-31 1013.9131 990.7177
fitHomeRuns      0.3466911 4.798143e-33 1002.2721 968.8202
fitRunsBattedIn  0.4451365 5.582361e-45 923.6755 840.4785
```

Figura 28: comparación de RMSE para lm y knn

Si observamos la comparación de los valores de RMSE para lm y RMSE para knn vemos como para knn obtenemos valores de RMSE más bajos que en lm.

A continuación, vamos a pasar a estudiar knn pero con cross-validation con las funciones facilitadas (modificadas) en las transparencias de este curso.

Como resultado de todos los modelos para knn tenemos:

```
> dfknn
      MSEtrain  MSEtest      RMSE
model1    702614 1018319 1007.936
model2   742710.3 1104580 1046.042
model3   948152.8 1344401 1155.095
model4   890438.6 1124682 1058.37
model5    701626 1070498 1032.528
model6   202087.6 566112.6 749.7489
model7   192618 548601.8 737.9448
model8   194167 537368.4 730.1435
model9   186634 518753.8 716.4729
model10  209475.1 517053.3 713.345
model11  199781.7 485465.5 694.0003
model12  185002.3 570724.3 754.1578
model13  246563.1 723447.4 849.7387
model14  185002.3 570724.3 754.1578
model15  185002.3 570724.3 754.1578
model16  180933.2 546368.9 736.5018
model17  176196.9 509648.3 711.9367
model18  212585.6 495377.6 699.6065
model19  478349.6 1096899 1045.575
model20  245229.6 533396.5 727.1972
model21  244354.7 502378.7 704.3497
model22  252035.7 467704.2 678.8639
model23  185212.9 502867.6 707.9723
model24  176511.7 515858.5 716.2531
model25  171678.8 510035.6 712.1532
model26  173701.2 533619.3 729.7357
```

Figura 29: comparación de los modelos para knn

Vemos los resultados obtenidos para todos los modelos knn. Estos modelos son los mismos que en lm, ya que para una correcta comparación tenemos que tener los mismos modelos.

Como podemos observar el que menor MSE en test obtiene es el modelo 22. El que menor MSE en train obtiene es el 25. El modelo que mayor MSE en test y en train presenta es el modelo 3. Por lo tanto, vamos a comparar los modelos 22 y 25 para ver cuál sería el mejor modelo de knn:

```
> dfknn[c(22,25),]
      MSEtrain  MSEtest      RMSE
model22  252035.7 467704.2 678.8639
model25  171678.8 510035.6 712.1532
```

Figura 30: comparación de los modelos para knn

Por lo tanto, para knn, elegiría como mejor modelo el modelo 22 al obtener un valor más pequeño de MSE en test. Este modelo sería:

```
> modelo22
Y ~ I(X8 * X13^2) + I(X8 * X15^2) + I(X11 * X13^2)
```

Figura 31: modelo22

3.4 Comparación de algoritmos

Para la comparación de algoritmos debo elegir entre el mejor modelo de lm o de knn. Estos modelos han sido el modelo 20 en lm y el 22 en knn:

```
> dflm[20,]
   EER T1    R2 T2 R2_Adj T3 MSEtrain MSEtest RMSE
model20 50.5 No 0.747 No 0.742 Si 388618 412666.4 625.4
> dfknn[22,]
   MSEtrain MSEtest RMSE
model22 252035.7 467704.2 678.8639
```

Figura 32: comparación de los modelos de lm y knn

De entre estos dos modelos elegiré como final en las comparaciones el modelo 20 de lm, ya que obtiene el valor de MSE en test más bajo de los dos.

Primeramente, para la comparación de los algoritmos se cargarán los datos de los archivos “regr_train_alumnos” y “regr_test_alumnos” para sustituir nuestros valores obtenidos de MSE del modelo 20.

El test de Wilcoxon nos proporciona los siguientes valores para test:

R+	R-	P-value
81	90	0.8650436

Donde obtenemos un p-value mayor a 0.05 no existiendo diferencias significativas entre ambos. Solo tenemos un 13,49% de confianza de que existan diferencias. Por lo tanto, no somos capaces a rechazar la hipótesis nula y concluiremos que no hay suficiente evidencia en los datos. Por lo tanto, aceptamos la hipótesis nula: las dos distribuciones son iguales.

El test de Wilcoxon nos proporciona los siguientes valores para train:

R+	R-	P-value
10	161	0.000328064

Los datos obtenidos en el train en el test de Wilcoxon muestra un p-value menor que el nivel de significación 0.05, por lo que el p-value es significativo. Entonces rechazaremos la hipótesis nula y decimos que las distribuciones son distintas. Esta interpretación tiene menor peso que la realizada anteriormente para test, ya que en el training los resultados son con datos ya conocidos por lo que se puede decir que hay sobre-ajuste para cada resultado de las bases de datos y que sean distintas las distribuciones entre los ajustes lm y knn.

Si realizamos el test de Friedman para test obtenemos un p-value del 0.0302 lo que quiere decir que existen diferencias significativas en al menos entre un par de algoritmos.

Como resultado del post-Hoc de Holm obtenemos:

1	2	
2	0.580	-
3	0.081	0.458

Figura 33: resultado de Post-Hoc de Holm

Por lo que existen diferencias significativas a favor de M5', mientras que lm vs knn pueden ser considerados equivalentes.

Si realizamos el test de Friedman para train obtenemos un p-value del 8.365e-05 lo que quiere decir que existen diferencias significativas en al menos entre un par de algoritmos.

Como resultado del post-Hoc de Holm obtenemos:

1	2
2	0.0031 -
3	0.0208 0.0056

Figura 34: resultado de Post-Hoc de Holm

Por lo que existen diferencias significativas a favor de los 3 algoritmos comparados. Lo que sugiere un sobre-ajuste.

4 Clasificación

Para la base datos “australian” voy a aplicar clasificación, de forma que vamos a obtener diferentes modelos como son, kNN (k nearest neighbor), LDA (linear discriminant analysis) y QDA (quadratic discriminant analysis), donde finalmente realizaremos una comparativa para ver cual se adapta mejor al problema.

4.1 k-NN

Tras realizar un breve análisis, se ha podido observar que para las diferentes clases hay 383 datos de la clase 0 y 307 de la clase 1. Es decir, un 55,5% de clase 0 y un 44,5% de clase 1.

Si normalizamos las variables y realizamos su visualización obtenemos:

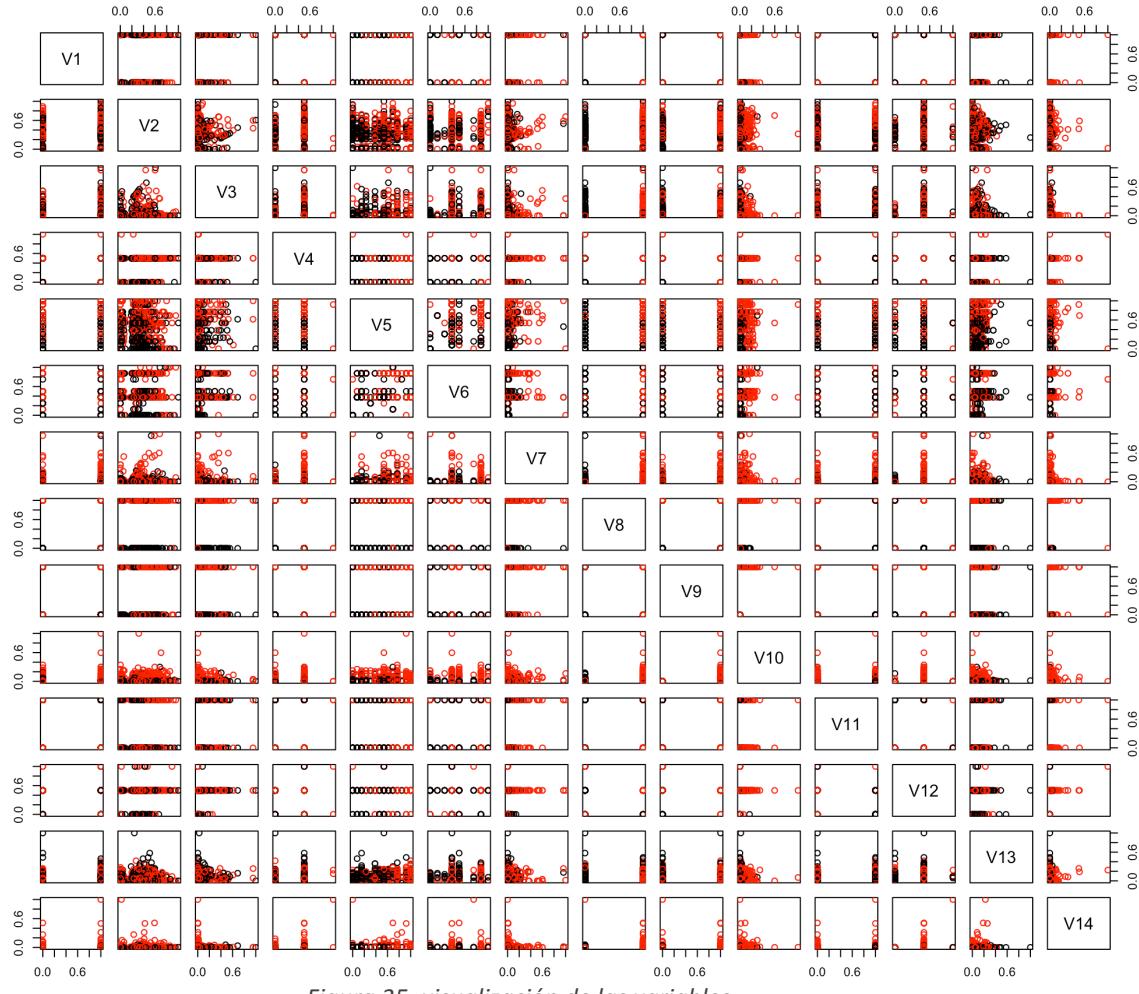


Figura 35: visualización de las variables

Si observamos la correlación entre las variables, podemos observar como la V9 y V10 son las que mayor correlación tiene, obteniendo un 57%. Tendremos en cuenta esto para un posterior modelo.

Comenzando los modelos, he hecho, primeramente, un proceso semejante al de las transparencias, por lo que he dividido el conjunto en train y test con un 80-20%. He realizado un knn con el conjunto de train, y un k =21, obteniendo que, a la hora de clasificar el test, se obtiene un modelo con un 85,5% de explicación del problema. Si cambio el k a 7, el modelo obtiene un 84,7% de explicación.

Si hacemos uso de caret, obtenemos que el mejor modelo es con k=9 y que su explicación sería del 84%.

Esto ha sido sobre los datos sin normalizar, por lo que si se hace normalizando los datos obtendríamos, con caret, un modelo que con k = 15, es el mejor favor obteniendo una explicación del 82,6%.

Una vez realizada esta prueba, he creado las funciones para realizar, de forma automática, el estudio de los distintos modelos. Estas funciones están inspiradas en las vistas en las transparencias de clase, usando los conjuntos de train y test que han sido dados junto con los dataset y usando caret en todos los modelos para obtener buenos valores medios.

En los apartados siguientes, concretamente en QDA, explicaré que modelos he usado y los resultados proporcionados por los diferentes métodos para no repetir la misma información en todos los apartados.

4.2 LDA

Antes de realizar LDA, he realizado, ya que se vio en clase, el modelo GLM obtenido, primeramente, con la división realizada por mí, un modelo que explica el 87,68% del problema. En el siguiente apartado realizaré la comparativa también con este modelo.

Para LDA, si realizo lo mismo que para kNN y GLM, obtengo un modelo que explica el 87,68% del problema. Y como en los casos anteriores, en el siguiente apartado mostraré la comparativa con el resto de modelo.

4.3 QDA

Por último, para QDA he realizado los mismos pasos que para el resto de modelos, obteniendo sobre mi conjunto de train y test un 78,26% de explicación del problema.

Una vez realizados todos los modelos, he pasado a compararlos, obteniendo la siguiente comparativa:

```
> resultados_clasificacion0
      knn      glm      lda      qda
Train 0.8737097 0.8791935 0.8603226 0.8064516
Test  0.8558824 0.8176471 0.8514706 0.7676471
Figura 36: comparativa de distintos modelos
```

En esta comparativa de la Figura 36, se pueden ver los distintos valores que se han obtenido para los modelos kNN, GLM, LDA y QDA.

Viendo estos resultados, he pasado a realizar distintos modelos de nuestro problema, haciendo uniones de variables o eliminando algunas de ellas que podrían ser innecesarias. Uno de los problemas que he tenido, ha sido la poca información disponible del dataset, ya que las variables no tienen ninguna información asociada y no se puede hacer ninguna suposición sobre ellas.

Por lo tanto, los distintos modelos que he realizado han sido:

- Modelo 0: todas las variables
- Modelo 1: Unión de las variables V9 y V10 ya que se vio que guardaban correlación entre ellas.
- Modelo 2: Al unir V9 con V10, se ha visto que la siguiente correlación más fuerte han sido las variables V5 y V6, por lo que se han unido también.
- Modelo 3: He realizado un GLM con todas las variables, de este nuevo data set, observando los p-value. Con este p-value voy a decidir eliminar las variables que tienen este valor mayor que 0.5. Estas variables son V1 y V3.
- Modelo 4: Para este modelo, he realizado otro GLM con las variables restantes y he decidido eliminar V2, V11, V12 y V7.
- Modelo 5: Con las correlaciones del dataset actual, voy a unir las variables V8 y V9, donde V9 era la unión de V9 y V10.

Una vez probados todos los modelos, kNN, GLM, LDA y QDA, he obtenido los siguientes resultados:

```
> resultados_clasificacion
   knn      glm      lda      qda      knn1      glm1      lda1      qda1      knn2      glm2      lda2      qda2
Train 0.8737097 0.8791935 0.8603226 0.8064516 0.8604839 0.8777419 0.8603226 0.7979032 0.8595161 0.8733871 0.8577419 0.7953226
Test  0.8558824 0.8176471 0.8514706 0.7676471 0.8544118 0.8073529 0.8514706 0.7661765 0.8500000 0.8058824 0.8441176 0.7632353
      knn3      glm3      lda3      qda3      knn4      glm4      lda4      qda4      knn5      glm5      lda5      qda5
Train 0.8653226 0.8735484 0.8572581 0.7841935 0.8750000 0.8746774 0.8572581 0.7793548 0.7906452 0.8085484 0.7785484 0.7680645
Test  0.8573529 0.8102941 0.8441176 0.7647059 0.8676471 0.8308824 0.8455882 0.7691176 0.7823529 0.7794118 0.7735294 0.7852941
```

Figura 37: comparativa de distintos modelos

De donde podemos ver como knn4 es el modelo que tiene un mayor valor en test. El mejor modelo para LDA es el modelo 0. Para GLM es el modelo 4. Para QDA el mejor modelo es el 5.

A continuación, pasaremos a realizar la comparación de los algoritmos.

4.4 Comparación de algoritmos

Para la comparación de los algoritmos he elegido los mejores de cada modelo, es decir, el mejor de kNN, LDA y QDA.

Primeramente, para la comparación de los algoritmos se cargarán los datos de los archivos “clasif_test_alumnos.csv” y “clasif_train_alumnos.csv” para sustituir nuestros valores obtenidos.

El test de Wilcoxon nos proporciona los siguientes valores para test:

El test de Wilcoxon nos proporciona los siguientes valores para test:

R+	R-	P-value
113	97	0.7841263

Donde obtenemos un p-value mayor a 0.05 no existiendo diferencias significativas entre ambos. Solo tenemos un 21,58 % de confianza de que existan diferencias. Por lo tanto, no somos capaces a rechazar la hipótesis nula y concluiremos que no hay suficiente evidencia en los datos. Por lo tanto, aceptamos la hipótesis nula: las dos distribuciones son iguales.

El test de Wilcoxon nos proporciona los siguientes valores para train:

R+	R-	P-value
144	66	0.1536465

Al igual que en test, para el train, no existen diferencias significativas al ser p-value mayor que 0.05.

Si realizamos el test de Friedman para test obtenemos un p-value del 0.9512 lo que quiere decir que no existen diferencias significativas en los algoritmos.

Como resultado del post-Hoc de Holm obtenemos:

1	2	
2	1	-
3	1	1

Figura 38: resultado de Post-Hoc de Holm

Si realizamos el test de Friedman para train obtenemos un p-value del 0.522 lo que quiere decir que no existen diferencias significativas los algoritmos.

Como resultado del post-Hoc de Holm obtenemos:

1	2	
2	0.66	-
3	0.66	0.53

Figura 39: resultado de Post-Hoc de Holm