



ugr

Universidad  
de Granada

Sistemas Inteligentes para la Gestión en la Empresa

# Práctica 1

---

**Pre-procesamiento de datos y clasificación binaria**

**Autor**

María Victoria Santiago Alcalá



Escuela Técnica Superior de Ingenierías Informática  
y de Telecomunicación

—  
Granada, Mayo de 2018

# Índice de contenidos

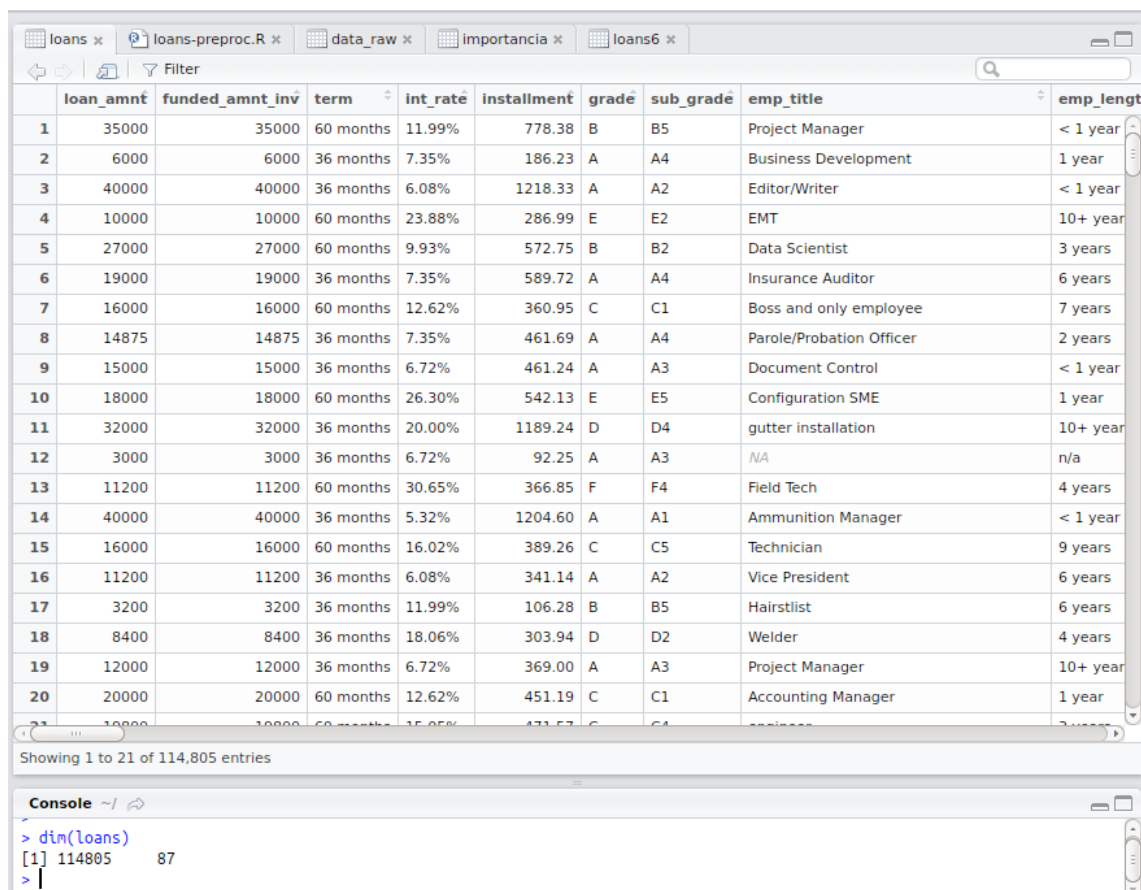
1. Exploración de datos.....	3
2. Preprocesamiento de datos.....	7
3. Técnicas de clasificación y discusión de resultados.....	21
4. Conclusiones.....	33
Bibliografía.....	34

# 1. Exploración de datos

Inicialmente se ha de cargar nuestro dataset LoanStats en R studio para llevar a cabo la comprobación de las variables que contiene comprobando también el listado proporcionado en el fichero excel (Data Dictionary) proporcionado para la realización de la práctica.

Por lo que una vez cargado en R podemos ver la dimesion que tiene nuestro dataset, viendo con ello el numero de filas y columnas. Tenemos 145 variables y 118650 entradas de las cuales eliminando valores nulos y desproporcionalidades nos quedará un dataset con el que se pueda trabajar.

A continuación se muestra nuestro dataset en una de las etapas en las que se han eliminado parte de las variables que no aportaban demasiada información. Se verá con mayor profundidad en el siguiente apartado.



	loan_amnt	funded_amnt_inv	term	int_rate	installment	grade	sub_grade	emp_title	emp_lengt
1	35000	35000	60 months	11.99%	778.38	B	B5	Project Manager	< 1 year
2	6000	6000	36 months	7.35%	186.23	A	A4	Business Development	1 year
3	40000	40000	36 months	6.08%	1218.33	A	A2	Editor/Writer	< 1 year
4	10000	10000	60 months	23.88%	286.99	E	E2	EMT	10+ year
5	27000	27000	60 months	9.93%	572.75	B	B2	Data Scientist	3 years
6	19000	19000	36 months	7.35%	589.72	A	A4	Insurance Auditor	6 years
7	16000	16000	60 months	12.62%	360.95	C	C1	Boss and only employee	7 years
8	14875	14875	36 months	7.35%	461.69	A	A4	Parole/Probation Officer	2 years
9	15000	15000	36 months	6.72%	461.24	A	A3	Document Control	< 1 year
10	18000	18000	60 months	26.30%	542.13	E	E5	Configuration SME	1 year
11	32000	32000	36 months	20.00%	1189.24	D	D4	gutter installation	10+ year
12	3000	3000	36 months	6.72%	92.25	A	A3	NA	n/a
13	11200	11200	60 months	30.65%	366.85	F	F4	Field Tech	4 years
14	40000	40000	36 months	5.32%	1204.60	A	A1	Ammunition Manager	< 1 year
15	16000	16000	60 months	16.02%	389.26	C	C5	Technician	9 years
16	11200	11200	36 months	6.08%	341.14	A	A2	Vice President	6 years
17	3200	3200	36 months	11.99%	106.28	B	B5	Hairstlist	6 years
18	8400	8400	36 months	18.06%	303.94	D	D2	Welder	4 years
19	12000	12000	36 months	6.72%	369.00	A	A3	Project Manager	10+ year
20	20000	20000	60 months	12.62%	451.19	C	C1	Accounting Manager	1 year
21	10000	10000	60 months	15.05%	473.57	C	C4	Business	2 years

Showing 1 to 21 of 114,805 entries

```
> dim(loan)
[1] 114805 87
>
```

Ilustracion 1. Dimensión del dataset tras eliminación de algunas variables.

También inicialmente, para tener una idea mas o menos de los datos, podemos ver la estructura interna que tiene:

```
> str(loans)
Classes 'tbl_df', 'tbl' and 'data.frame':    114805 obs. of  87 variables:
 $ loan_amnt      : int  35000 6000 40000 10000 27000 19000 16000 14875 15000 18000 ...
 $ funded_amnt_inv : int  35000 6000 40000 10000 27000 19000 16000 14875 15000 18000 ...
 $ term           : chr  "60 months" "36 months" "36 months" "60 months" ...
 $ int_rate       : chr  "11.99%" "7.35%" "6.08%" "23.88%" ...
 $ installment    : num  778 186 1218 287 573 ...
 $ grade          : chr  "B" "A" "A" "E" ...
 $ sub_grade      : chr  "B5" "A4" "A2" "E2" ...
 $ emp_title      : chr  "Project Manager" "Business Development" "Editor/Writer" "ENT" ...
 $ emp_length     : chr  "< 1 year" "1 year" "< 1 year" "10+ years" ...
 $ home_ownership : chr  "RENT" "RENT" "MORTGAGE" "RENT" ...
 $ annual_inc     : num  65000 65000 60000 42000 68000 ...
 $ verification_status : chr  "Source Verified" "Not Verified" "Source Verified" "Source Verified" ...
 $ issue_d        : chr  "Dec-2017" "Dec-2017" "Dec-2017" "Dec-2017" ...
 $ loan_status    : chr  "Paid" "Paid" "Paid" "Paid" ...
 $ purpose        : chr  "debt_consolidation" "debt_consolidation" "debt_consolidation" "credit_card" ...
 $ zip_code       : chr  "926xx" "070xx" "200xx" "726xx" ...
 $ addr_state     : chr  "CA" "NJ" "DC" "AR" ...
 $ dti            : num  26.81 8.35 25.46 31.17 30.04 ...
 $ delinq_2yrs    : int  0 0 0 0 0 0 1 0 0 ...
 $ earliest_cr_line : chr  "Oct-2005" "Oct-1998" "Aug-2000" "Oct-1998" ...
 $ inq_last_6mths  : int  0 1 0 0 0 0 0 0 0 ...
 $ open_acc       : int  5 6 4 14 14 9 10 11 5 8 ...
 $ pub_rec        : int  0 0 0 0 0 0 1 0 0 ...
 $ revol_bal      : int  21026 4348 37523 34122 19008 11958 9455 14642 15474 18269 ...
 $ revol_util     : chr  "63.7%" "17.7%" "64.7%" "79.5%" ...
 $ total_acc      : int  16 23 6 20 20 29 16 36 19 12 ...
 $ initial_list_status : chr  "w" "w" "w" "w" ...
 $ out_prncp      : num  34571 5851 38984 9912 25444 ...
 $ out_prncp_inv   : num  34571 5851 38984 9912 25444 ...
 $ total_pymnt    : num  720 180 1185 254 1736 ...
 $ total_pymnt_inv : num  720 180 1185 254 1736 ...
 $ total_rec_prncp : num  429 149 1016 88 1556 ...
 $ total_rec_int   : num  291.4 30.6 168.9 165.8 179.9 ...
 $ last_pymnt_d    : chr  "Feb-2018" "Feb-2018" "Feb-2018" "Feb-2018" ...
 $ last_pymnt_amnt : num  778 186 1218 287 573 ...
 $ last_credit_pull_d : chr  "Feb-2018" "Feb-2018" "Dec-2017" "Feb-2018" ...
```

Ilustración 2. Estructura interna

Y a continuación vemos la distribución de los valores de su variable clase:

```
> prop.table(table(loans$loan_status))

    Not_paid      Paid
0.01534776 0.98465224
```

Ilustración 3. Variable loans\_status

Con ello se aprecia que son pagados el 98.46% de ellos. Por ello como modelo inicial se podría hacer la predicción de que todos finalmente son pagados.

Al ir explorando el dataset nos damos cuenta de que tenemos variables con casos muy desequilibrados como por ejemplo ocurre con la variable `hardship_flag` la cual podemos ver en el siguiente gráfico.

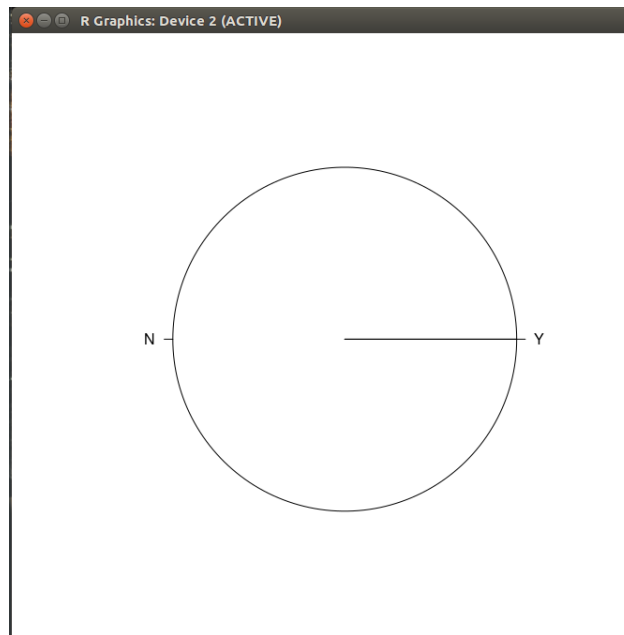


Ilustración 4. Ejemplo de desequilibrio en los casos de la variables

En la siguiente ilustración también se puede apreciar el desequilibrio entre los distintos valores de la variable:

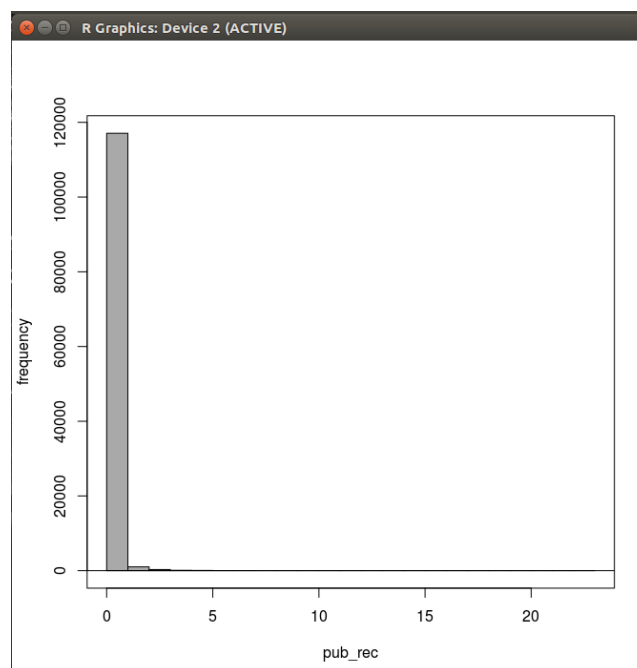


Ilustración 5. Ejemplo de desequilibrio en sus valores

Al igual también vemos que hay casos en los que la variable tiene solo un único valor por lo que no nos aporta tampoco mucha información relevante.

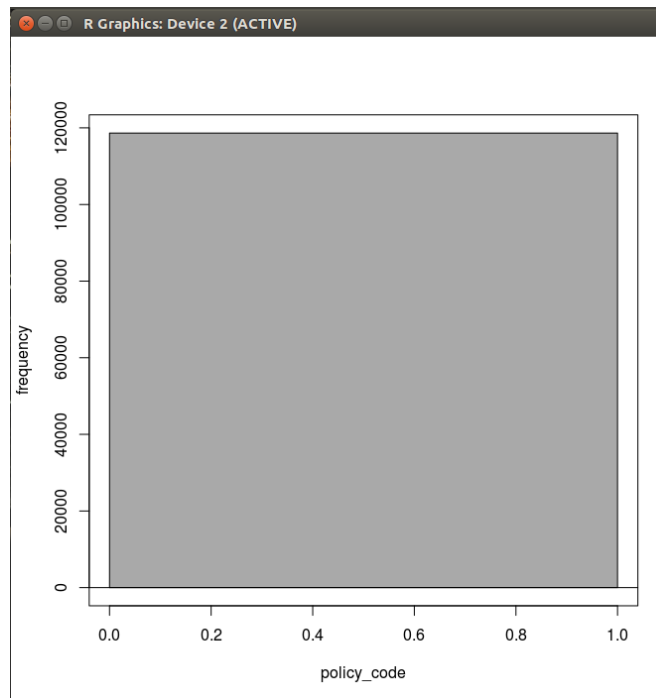


Ilustración 6. Ejemplo de variable con un solo valor en sus casos

En el análisis exploratorio de las variables se puede ir más allá, por ejemplo se pueden ver las relaciones entre variables como por ejemplo en el siguiente caso entre la cantidad del préstamo y las cuentas que e han abierto.

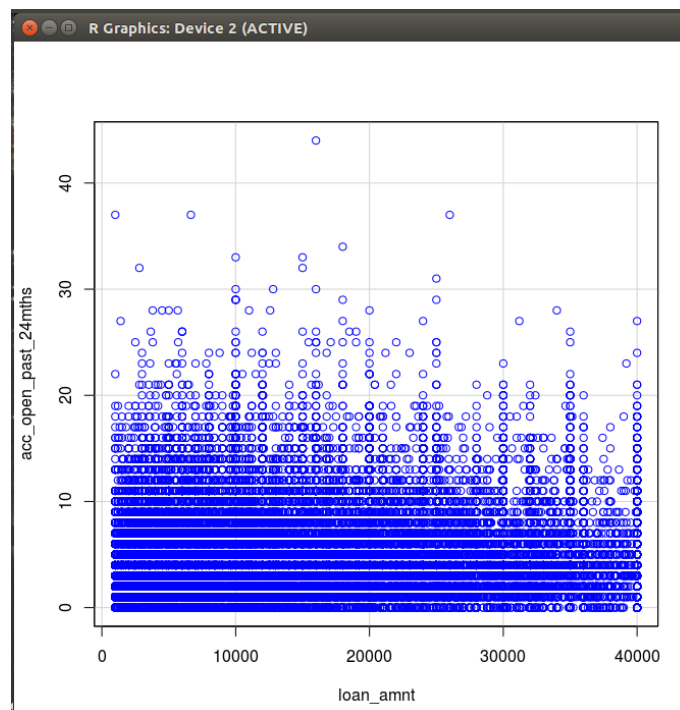


Ilustración 7. Diagrama de las variables acc\_open\_past\_24mths y loan\_amnt

## 2. Preprocesamiento de datos

### ➤ Eliminación de instancias pertenecientes a clases no relevantes

El problema en si consiste en llevar a cabo una predicción del estado de un préstamo el cual viene representado por la variable `loan_status`, a partir del resto de variables.

Por lo que el primer paso realizado ha sido la normalización de la clase variable eliminando y transformando los datos correspondientes a cada fila puesto que se especifica el conjunto de datos como un problema de clasificación binaria con dos únicos tipos de salida: Pago o Impago. Por lo que se han reducido los datos a estas dos posibles salidas.

```
# Normalize class variable
```

```
>class_variable = sapply(loans$loan_status, function(column_value) {  
  
  if(column_value %in% c("Current")){  
  
    "Paid"  
  
  }else if(column_value %in% c("Late (31-120 days)", "Late (16-30 days)", "In Grace Period", "Charged Off")){  
  
    "Not_paid"  
  
  }else{  
  
    "To be removed"  
  
  }  
  
})  
  
>loans$loan_status <- class_variable  
  
>loans <- loans[loans$loan_status != "To be removed", ]
```

A continuación, se ha seguido con el proceso de eliminación de instancias pertenecientes a clases no relevantes.

## ➤ Eliminación de variables sin información

Se ha procedido a comprobar los valores perdidos creando una variable en R con el contenido de todos los nombres de nuestros atributos(columnas) del dataset. Para ello se ha usado la función `is.na()`.

Seguidamente se ha agrupado cada atributo en función del porcentaje de valores perdidos que se tiene creando tres grupos: valores perdidos con un porcentaje al menos del: 75%, 50% y 25%; donde se han eliminado aquellos que tienen el 75% y el 50%.

Visualizando de nuevo nuestro dataset vemos que las 3 primeras columnas parecen iguales pero al aplicar la función “identical” vemos que las dos que son iguales son las dos primeras mientras que la tercera tiene valores distintos.

```
>identical(loans$loan_amnt, loans$funded_amnt)
```

Returns -> True

```
>identical(loans$funded_amnt, loans$funded_amnt_inv)
```

Returns -> False

Así que sabiendo que las dos primeras son iguales pasamos a eliminar una de ellas:

```
loans <- subset(loans, select= -funded_amnt)
```

Continuamos explorando el contenido de las siguientes columnas y procedemos a eliminar las siguientes ya que no aportan información debido a la diversidad de sus valores.

Valores muy desequilibrados:

Ejemplo con valores S, N, 0, 4, 13 y 26

S->114600 N->3 ó 0->114690, 4->1, 13->2, 26->1

- `table(loans$hardship_flag)`: Donde se puede apreciar que todas las filas no tienen este plan en sus préstamos y solo un caso si  
N=114804 Y=1
- `table(loans$debt_settlement_flag)`: Ocurre lo mismo que con el anterior y recibimos el mismo resultado



- num\_tl\_30dpd el cual muestra el siguiente output:  
0->114756    1->46    2->3  
Por lo que se puede ver de nuevo que hay un gran desequilibrio.
- num\_tl\_120dpd\_2m: Se elimina por el gran desequilibrio que se ha encontrado.
- delinq\_amnt: Eliminada también por el desequilibrio.
- chargeoff\_within\_12\_mths: Como muestra el siguiente output también esta desequilibrada:  

```
>table(loans$chargeoff_within_12_mths)
```

	0	1	2	3	4	5
	113972	780	40	5	7	1
- acc\_now\_delinq:  

```
>table(loans$acc_now_delinq)
```

	0	1	2	3
	114727	74	3	1

```
>loans$acc_now_delinq <- NULL
```
- next\_pymnt\_d:  

```
>table(loans$next_pymnt_d)
```

	Apr-2018	Feb-2018	Mar-2018
	21	22	114734

```
>loans$next_pymnt_d <- NULL
```
- recoveries:  

```
>table(loans$recoveries)
```

	0	2970
	114804	1

```
>loans$recoveries <- NULL
```
- total\_rec\_late\_fee:  

```
>table(loans$total_rec_late_fee)
```

	0	15	15.04	15.06
	114350	166	1	2

```
>loans$total_rec_late_fee <- NULL
```

- pymnt\_plan:

```
>table(loans$pymnt_plan)
      n      y
114804    1
>loans$pymnt_plan <- NULL
```

Todas las variables anteriores como se ha podido apreciar en sus resultados no resultan relevantes por lo que se han eliminado por el gran desequilibrio que se tenía en sus distintos casos (como se vio inicialmente en el análisis exploratorio)

A continuación se presentan las variables que se han encontrado y eliminado debido a que solo tienen un único valor en todas sus filas por lo que tenían poca diversidad en sus valores

- policy\_code:

```
>table(loans$policy_code)
      1
114805
>loans$policy_code <- NULL
```

- collection\_recovery\_fee:

```
>table(loans$collection_recovery_fee)
      0
114805
>loans$collection_recovery_fee <- NULL
```

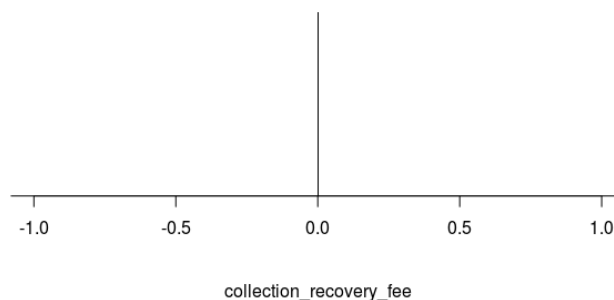


Ilustración 8. Valores de la variable collection\_recovery\_fee

Una vez eliminadas todas estas variables, si vemos la dimensión nueva de nuestro dataset tendremos que lo hemos reducido quedándonos 114805 filas y 88 variables.

```
>dim(loans)
114805  88
```

## ➤ Transformación

A continuación, pasamos a transformar por ejemplo los valores de la columna "title" para compararla con los de la columna "purpose".

Seleccionamos las dos columnas:

```
>title_purpose <- subset(loans, select=c("title", "purpose"))
```

Ahora mostramos el numero de elementos distintos que tienen ambas columnas:

```
>length(unique(title_purpose$title))
```

```
[13
```

```
>length(unique(title_purpose$purpose))
```

```
[1] 13
```

Mostramos las gráficas de sectores de ambos para compararlos (Ilustraciones 9 y 10).

```
>with(title_purpose, pie(table(title), labels=levels(title), xlab="", ylab="",  
main="title", col=rainbow_hcl(13)))
```

```
>with(title_purpose, pie(table(purpose), labels=levels(purpose), xlab="",  
ylab="", main="purpose", col=rainbow_hcl(13)))
```

También mostramos la tabla donde también podemos apreciar claramente que se corresponden los valores:

```
> table(loans$title)
```

Business	Car financing	Credit card refinancing
1455	1311	24677
Debt consolidation	Green loan	Home buying
60567	72	1345
Home improvement	Learning and training	Major purc
8341	1	3094
Medical expenses	Moving and relocation	Other
1872	862	10387
Vacation		
821		

```
>table(loans$purpose)
```

```

      car      credit_card debt_consolidation      educational
1311      24676      60568          1
home_improvement      house      major_purchase      medical
8341      1345      3093      1873
moving      other      renewable_energy      small_business
862      10387      72      1455
vacation
821

```

Por lo que si los comparamos vemos que son prácticamente iguales:

Comparación de variables			
Title		Purpose	
Business	1455	small_business	1455
Car financing	1311	car	1311
Credit card refinancing	24677	credit_card	24676
Debt consolidation	60567	debt_consolidation	60568
Green loan	72	renewable_energy	72
Home buying	1345	house	1345
Home improvement	8341	home_improvemen	8341
Learning and training	1	educational	1
Major purchase	3094	major_purchase	3093
Medical expenses	1872	medical	1873
Moving and relocation	862	moving	862
Vacation	821	vacation	821
Home buying	1345	house	1345
Other	10387	other	10387

Ilustración 9. Comparación de variables

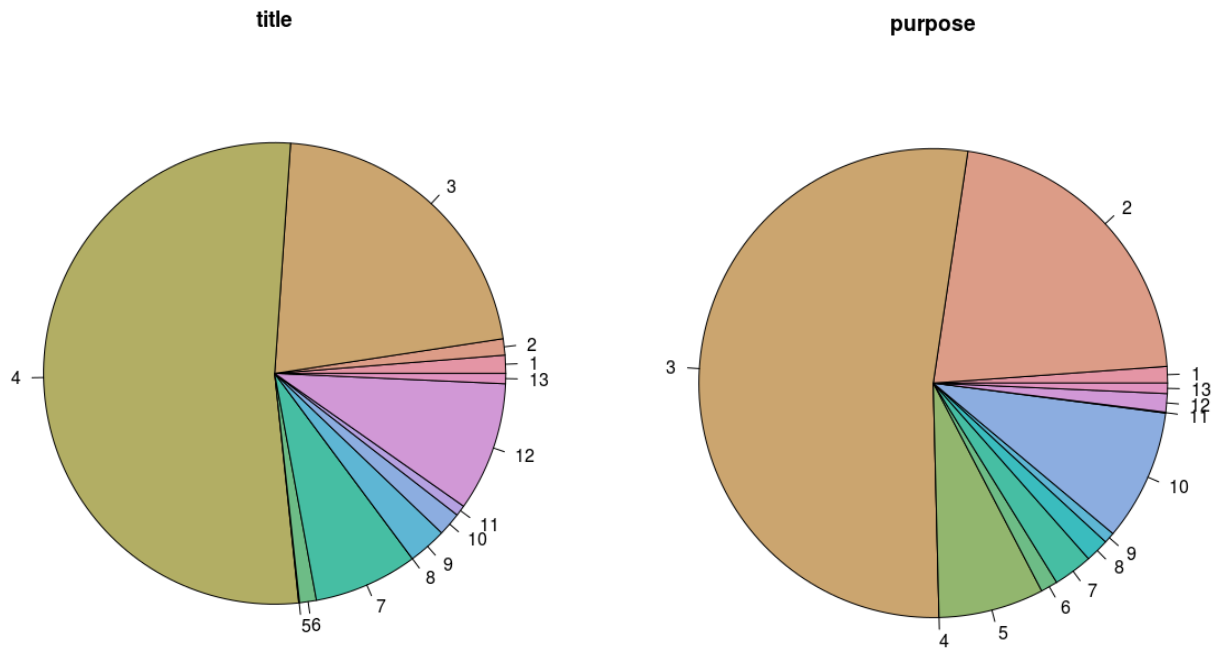


Ilustración 10. Comparación de las variables con gráfico de sectores

Finalmente eliminamos la columna title ya que solo se diferencia de en 2 valores con la columna purpose la cual tiene los valores transformados para su tratamiento mientras que los de la columna title no están en buen formato para su tratamiento.

Llegados a este punto, tenemos 87 variables por lo que para reducir este numero he optado por ver la correlación existente entre cada par de variables con el fin de llevar a cabo esa simplificación del dataset.

## ➤ Reducción de la dimensionalidad

- Correlación por cada par de variables con el método de Pearson.

Para llevar a cabo este proceso, primero vamos a dividir las variables según el tipo en dos grupos: valores continuos y valores categóricos.

1. Almacenamos las columnas cuyos valores son continuos:

```
>numeric_columns = cols[lapply(loans, typeof) != "character"]
```

2. Almacenamos las columnas cuyos valores son categóricos:

```
>character_columns = cols[lapply(loans, typeof) == "character"]
```

3. A continuación recorreremos por cada par de valores de los valores continuos con el fin de conocer su coeficiente de correlación obteniendo el siguiente resultado:

```
> View(pairs)
> print(pairs)
```

	col1	col2	coefi
1	loan_amnt	installment	9.999959e-01
2	loan_amnt	annual_inc	9.999942e-01
3	loan_amnt	dti	9.992125e-01
4	loan_amnt	delinq_2yrs	9.947725e-01
5	loan_amnt	inq_last_6mths	9.947554e-01
6	loan_amnt	open_acc	9.736190e-01
7	loan_amnt	pub_rec	9.704104e-01
8	loan_amnt	revol_bal	9.501817e-01
9	loan_amnt	total_acc	9.501446e-01
10	loan_amnt	out_prncp	9.493671e-01
11	loan_amnt	out_prncp_inv	9.451399e-01
12	loan_amnt	total_pymnt	9.317063e-01
13	loan_amnt	total_pymnt_inv	9.315545e-01
14	loan_amnt	total_rec_prncp	8.931569e-01
15	loan_amnt	total_rec_int	8.782348e-01
16	loan_amnt	last_pymnt_amnt	8.524337e-01
17	loan_amnt	collections_12_mths_ex_med	8.516426e-01
18	loan_amnt	tot_coll_amt	8.443971e-01
19	loan_amnt	tot_cur_bal	8.437770e-01
20	loan_amnt	open_acc_6m	8.411652e-01
21	loan_amnt	open_act_il	8.389085e-01
22	loan_amnt	open_il_12m	8.354959e-01
23	loan_amnt	open_il_24m	8.337521e-01
24	loan_amnt	mths_since_rcnt_il	8.245850e-01
25	loan_amnt	total_bal_il	8.145454e-01
26	loan_amnt	il_util	8.052920e-01
27	loan_amnt	open_rv_12m	7.938123e-01
28	loan_amnt	open_rv_24m	7.874085e-01
29	loan_amnt	max_bal_bc	7.868629e-01
30	loan_amnt	all_util	7.824673e-01
31	loan_amnt	total_rev_hi_lim	7.788090e-01
32	loan_amnt	inq_fi	7.743750e-01

Ilustración 11. Tabla con los pares de valores

Finalmente eliminamos las variables con coeficiente de correlación mayor o igual que 0,95.

val_cor x			
pairs x			
x[order("-coefi")] x			
x[order("-coefi"), ] x			
Filter			
	col1	col2	coefi
1	loan_amnt	installment	0.9999959
2	loan_amnt	annual_inc	0.9999942
3	loan_amnt	dti	0.9992125
4	loan_amnt	delinq_2yrs	0.9947725
5	loan_amnt	inq_last_6mths	0.9947554
6	loan_amnt	open_acc	0.9736190
7	loan_amnt	pub_rec	0.9704104
8	loan_amnt	revol_bal	0.9501817
9	loan_amnt	total_acc	0.9501446
10	loan_amnt	out_prncp	0.9493671
11	loan_amnt	out_prncp_inv	0.9451399
12	loan_amnt	total_pymnt	0.9317063
13	loan_amnt	total_pymnt_inv	0.9315545

Showing 1 to 13 of 1,431 entries

Ilustración 12. Valores de correlación ordenados decreciente-mente

Son las siguientes:

installment,  
annual\_inc,  
dti,  
delinq\_2yrs,  
inq\_last\_6mths,  
open\_acc,  
pub\_rec,  
revol\_bal,  
total\_acc

El proceso seguido para la correlación por cada par de variables ha sido el siguiente:

```
# Almacenamos las columnas cuyos valores son continuos
numeric_columns = cols[lapply(loans, typeof) != "character"]

# Almacenamos las columnas cuyos valores son categoricos
character_columns = cols[lapply(loans, typeof) == "character"]

# Recorremos por cada par de valores para ver su correlación
correlation_array <- c("i","j","cor_coef")

# Para ver mas codigo vayase a el script de R
```

Y finalmente lo almacenamos en “pairs”

### ➤ **Arboles de decisión**

Una vez eliminadas las variables anteriores se ha procedido a ejecutar arboles de decisión para ver también cuales son las variables mas usadas por lo tanto las mas importantes para seguir reduciendo la dimensionalidad de nuestro dataset. Se ha de notar que posteriormente volveremos a usar arboles de decisión para hacer la clasificación.

Se han usado el conjunto de arboles disponibles en el paquete `rpart`. Creamos nuestro árbol de decisión de la siguiente forma:

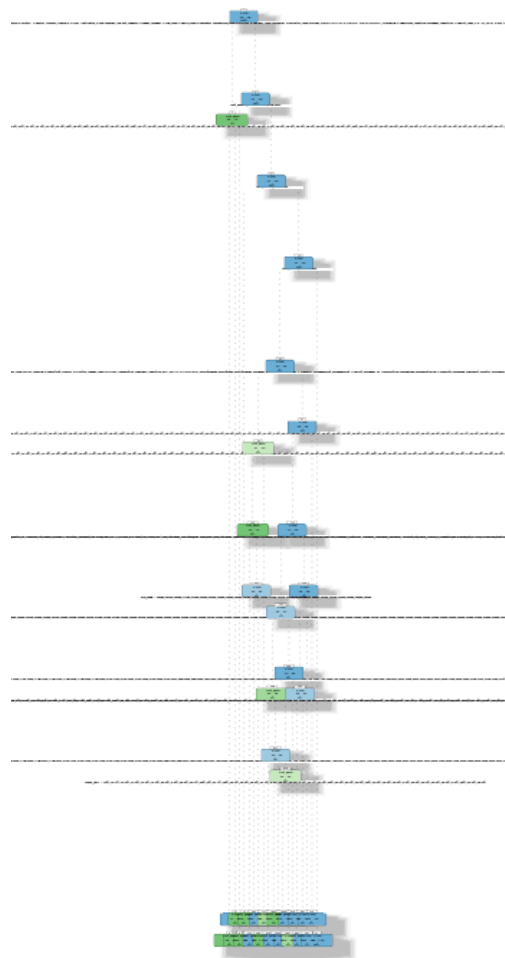
```
>fit <- rpart(loans$loan_status ~ ., data = loans, method = "class", control = list(maxdepth = 5))
```

Posteriormente procedemos a visualizarlo:

```
>fancyRpartPlot(fit)
```

En la siguiente imagen se muestra el resultado de como quedaría:





Rattle 2018-Apr-30 22:17:53 stiago

Ilustración 13. Árbol de decisión

Console ~/ ↻

Variable importance

emp_title	revol_util	last_pymnt_d	zip_code	earliest_cr_line	last_pymnt_amnt
27	11	11	9	9	6
total_rec_int	total_pymnt	total_pymnt_inv	total_rec_prncp	int_rate	addr_state
5	5	5	5	2	1
sub_grade	emp_length				
1	1				

Ilustración 14. Importancia

Si introducimos en nuestra consola `summary(fit)` podremos ver las variables más importantes como se puede visualizar en la imagen anterior.

## ➤ Random Forest

Se puede decir que es una mejora de los árboles de decisión la cual se ha usado para garantizar finalmente el número de variables que se han seleccionado.

Se caracteriza por estar formado por múltiples árboles de decisión individuales. En este paso únicamente lo usaremos para obtener las variables más importantes. Posteriormente en el proceso de clasificación se abordará más profundamente.

Para obtener el número de variables a usar, se han eliminado de nuestro dataset las filas de casos con NA valores reduciéndolo de 114805 filas a 73002.

Seguidamente, nos hemos quedado con los atributos numéricos de la siguiente forma:

```
>nums <- unlist(lapply(loans6, is.numeric))
>loans6.numeric = loans6[, nums]
>tmp <- cor(loans6.numeric)
>tmp[upper.tri(tmp)] <- 0
>diag(tmp) <- 0
```

A continuación se ha llevado a cabo la eliminación de variables por alta correlación:

```
>loans6.important.numeric <- loans6.numeric[,!apply(tmp,2,function(x)
any(x > 0.95))]
>head(loans6.important.numeric)
>names(loans6.important.numeric)
```

Seguidamente, he añadido nuestra variable clase I (a columna loan\_status) a nuestro actual dataset.

```
>loans6.important.numeric$loan_status <- loans6$loan_status
```

A continuación he aplicado la primera técnica de random forest:

```
>rf.model <- randomForest(loans6.important.numeric$loan_status ~.,
  data = loans6.important.numeric,
  ntree = 35,
  type="classification",
  importance=TRUE,
  na.action=na.omit)
```

Al mostrarla nos da el siguiente resultado:

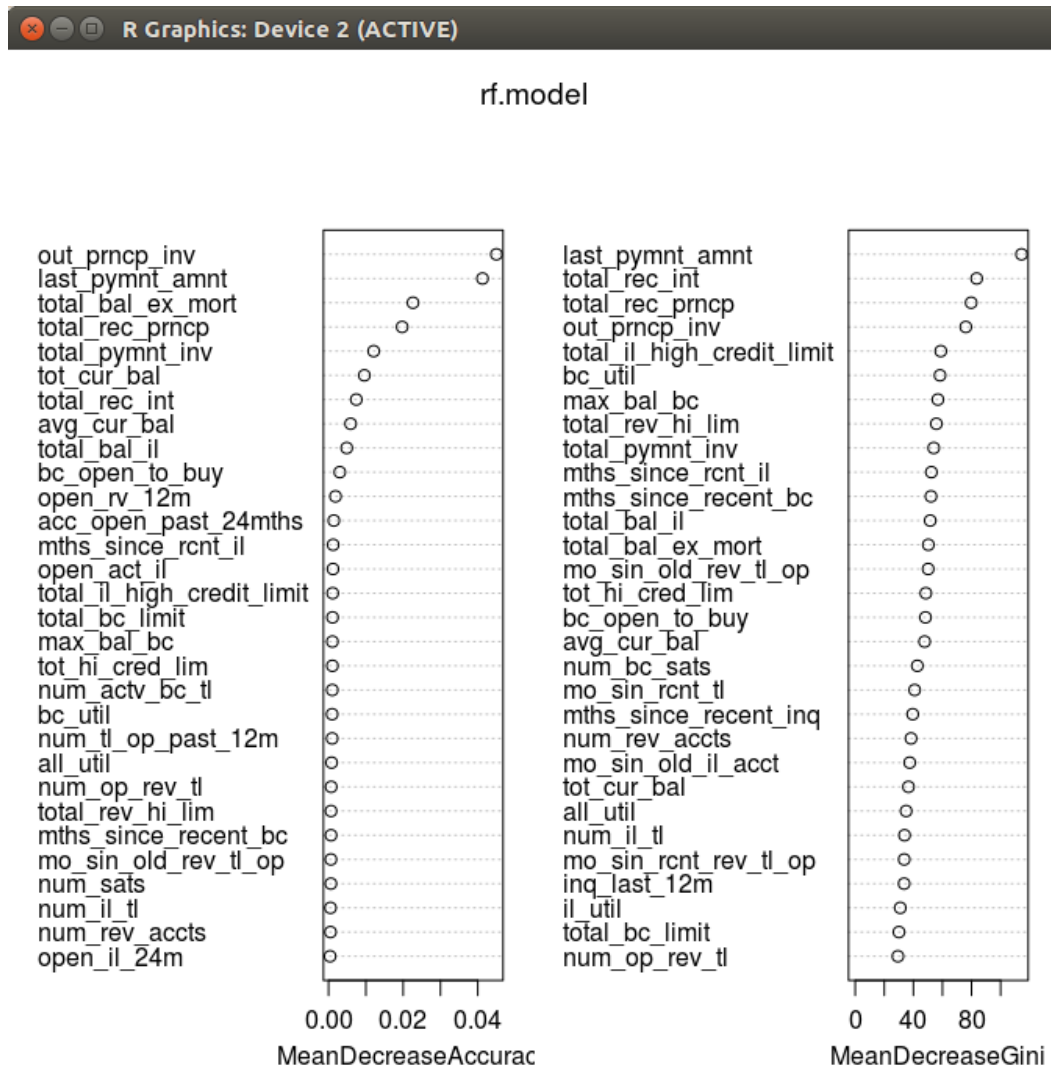


Ilustración 15. Primer modelo

En esta imagen se puede ver la medida MeanDecreaseGini la cual es una medida de desorden basada en que a mayor medida mayor importancia en los modelos creados puesto que valores próximos a cero van a implicar que hay un mayor desorden mientras que valores próximos a 1 se corresponden a un menor desorden.

	Not_paid	Paid	MeanDecreaseAccuracy	MeanDecreaseGini
num_il_tl	-2.62126379	8.001580883	7.89416348	37.648593
mths_since_rcnt_il	-0.92564137	8.462116315	7.76723322	39.520270
il_util	-2.00031904	6.652792571	6.46191291	42.214303
mths_since_recent_bc	0.32949618	7.721417571	7.75244773	42.965925
all_util	-2.31151073	6.015032753	5.96563089	43.359902
total_bal_ex_mort	-4.10803423	4.383415502	4.37331160	46.041031
total_bc_limit	-5.48123604	5.736416661	5.72533923	46.314910
bc_util	-2.83359110	7.098898056	7.07756135	46.836738
total_rev_hi_lim	-4.78138549	5.130587979	5.10630129	47.894408
total_il_high_credit_limit	-3.18924357	9.167324789	9.09515827	48.888245
tot_hi_cred_lim	-4.94854618	6.835759712	6.78547161	50.865626
mo_sin_old_rev_tl_op	0.17438103	5.506723019	5.56013004	51.080979
avg_cur_bal	-2.20467813	6.025094573	6.01369208	51.402528
bc_open_to_buy	-3.97018559	6.116650185	6.15111108	51.414695
mo_sin_old_il_acct	-0.07473492	4.678626169	4.71825870	51.586452
total_bal_il	-3.33399779	3.935466515	3.92939569	51.920323
max_bal_bc	-3.20288209	3.781077957	3.79074364	52.548155
total_pymnt_inv	-6.51478599	8.374754291	8.37206730	67.674126
total_rec_int	-7.04890517	7.818898462	7.78696232	67.734781
total_rec_prncp	-3.76967732	9.187598782	9.19856603	81.898095
out_prncp_inv	-5.47468421	9.084364648	9.11833502	85.247834
last_pymnt_amnt	-9.90328493	11.625375930	11.61130169	88.727515

Showing 31 to 52 of 52 entries

Console

```
> View(importancia)
> 
```

Ilustración 16. Tabla con los valores de las variables

Si ahora realizamos el mismo proceso con las variables categóricas tendremos el total de variables a analizar.

Finalmente nos hemos quedado con las siguientes:

loans\_status  
 emp\_title  
 purpose  
 verification\_status  
 revol\_util  
 last\_pymnt\_d  
 zip\_code  
 last\_pymnt\_amnt  
 total\_rec\_int  
 total\_pymnt  
 total\_pymnt\_inv  
 total\_rec\_prncp  
 int\_rate  
 sub\_grade  
 emp\_length

### 3. Técnicas de clasificación y discusión de resultados

Una vez tenemos el preprocesamiento del dataset, vamos a pasar a la fase de clasificación en la cual vamos a aplicar modelos de predicción como árboles de decisión(rpart), random forest o SVM.

Se ha dividido el dataset en dos partes, validación (30% del conjunto) y entrenamiento(70%).

Para ello se ha usado sample como muestra la siguiente captura en la cual podemos observar también la dimensionalidad de cada uno de los subsets.

```
> set.seed(1234)
>
> ind <- sample(2, nrow(final_loans), replace = TRUE, prob = c(0.7, 0.3))
> entrenamiento <- final_loans[ind == 1, ]
> validacion <- final_loans[ind == 2, ]
>
> dim(entrenamiento)
[1] 51015    15
> dim(validacion)
[1] 21987    15
> |
```

Ilustración 17. División del dataset

A continuación vamos a comenzar aplicando un árbol de decisión simple.

- Árboles de decisión

El árbol de decisión usado ha sido como en el anterior capítulo el proporcionado por el paquete rpart.

Para ello únicamente se ha tenido que lanzar el siguiente comando sobre el set de entrenamiento.

```
> my_form <- loans_status ~ emp_title + purpose + verification_status + revol_util +
last_pymnt_d + zip_code + last_pymnt_amnt + total_rec_int + total_pymnt +
total_pymnt_inv + total_rec_prncp + int_rate + sub_grade + emp_length
> fit_train <- rpart(my_form, data=entrenamiento, method="class")
```

Y a continuación solo queda ver el grafico que genera el árbol de decisión haciendo: fancyRpartPlot(fit\_train)

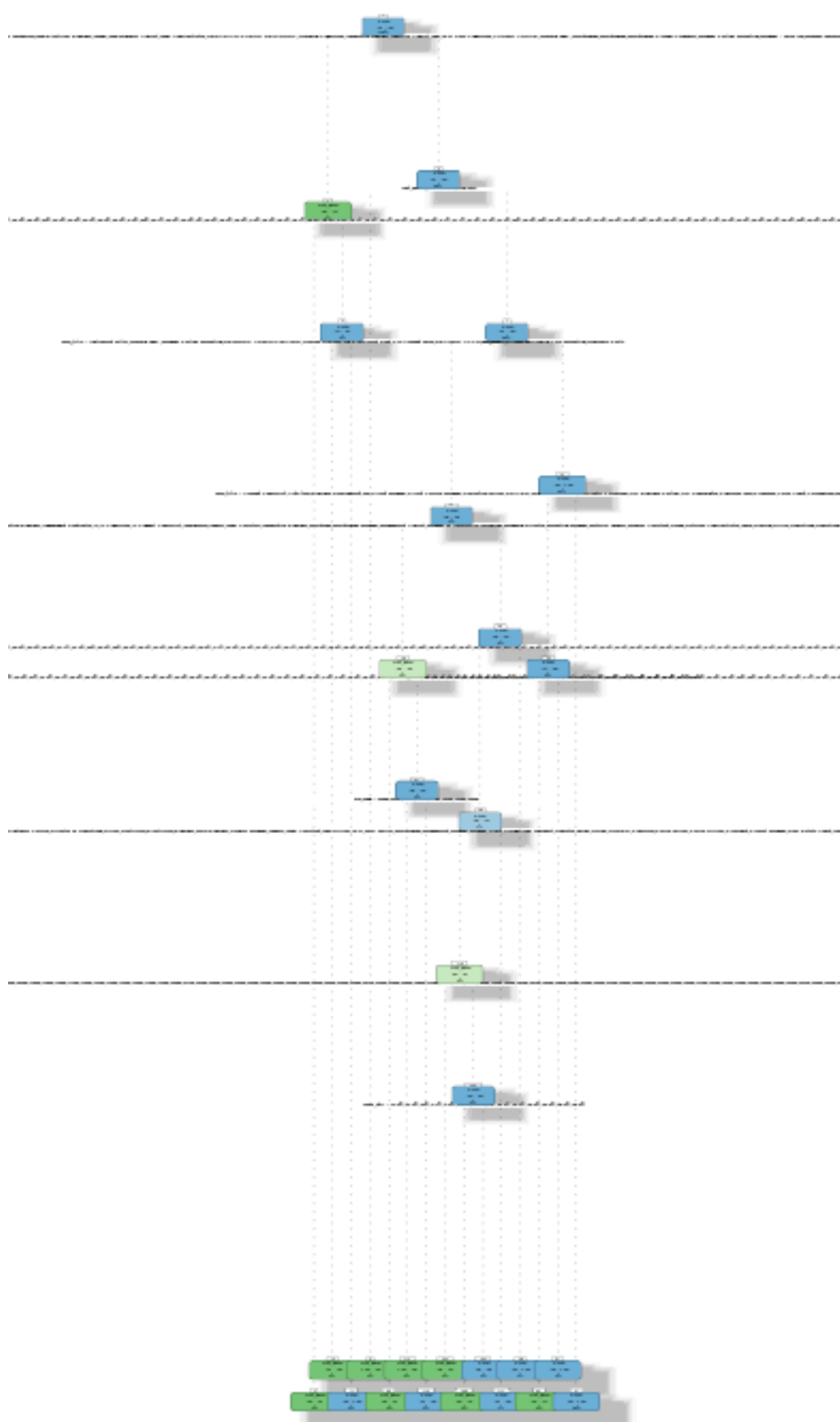


Ilustración 18. Árbol de decisión

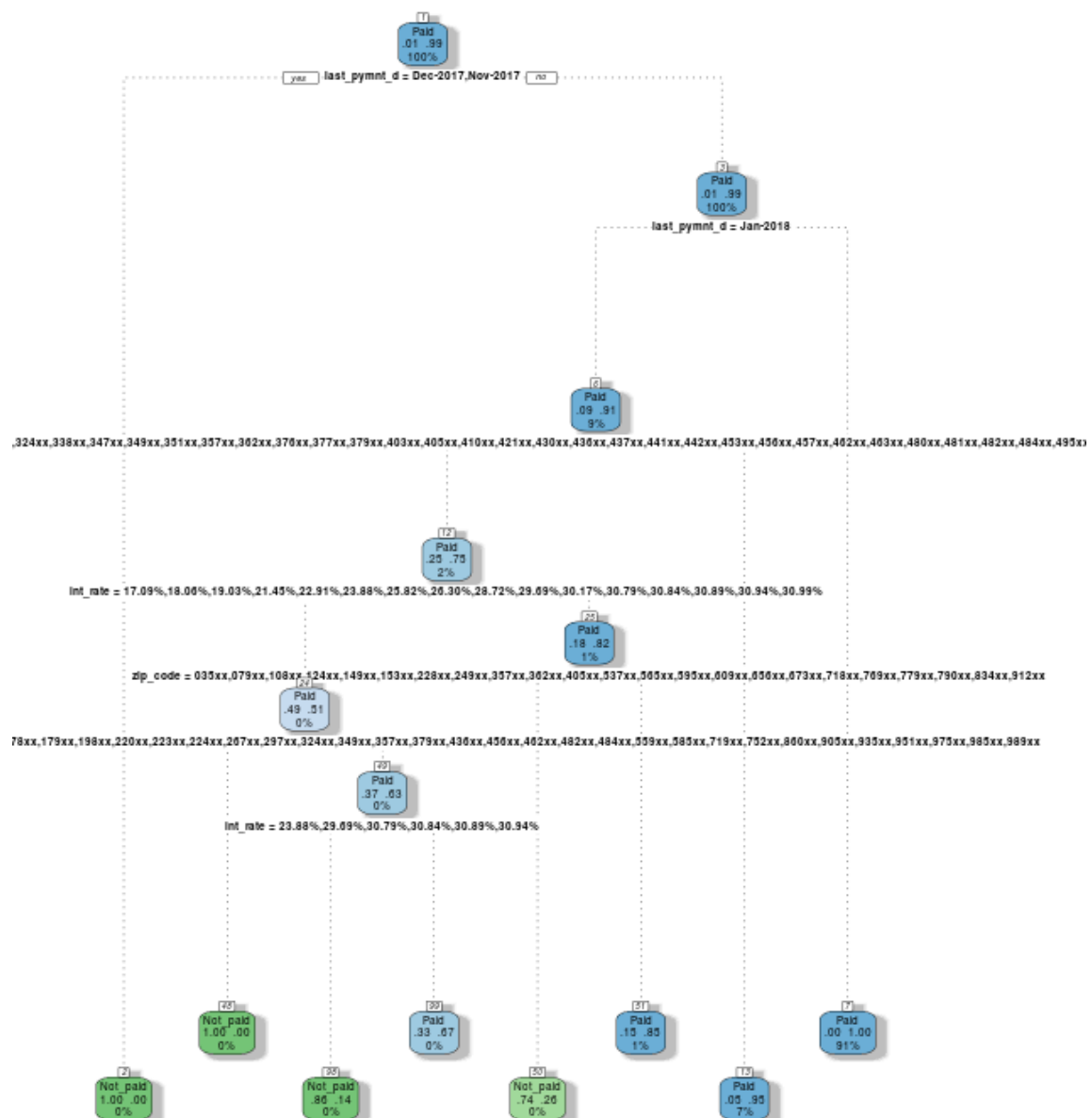


Ilustración 19. Árbol de decisión

- Roc:

Creamos ahora una curva ROC de un modelo predictivo de rpart sobre datos de test (validación) usando el paquete ROCR. Esto para medir eficiencia de predicción del modelo.

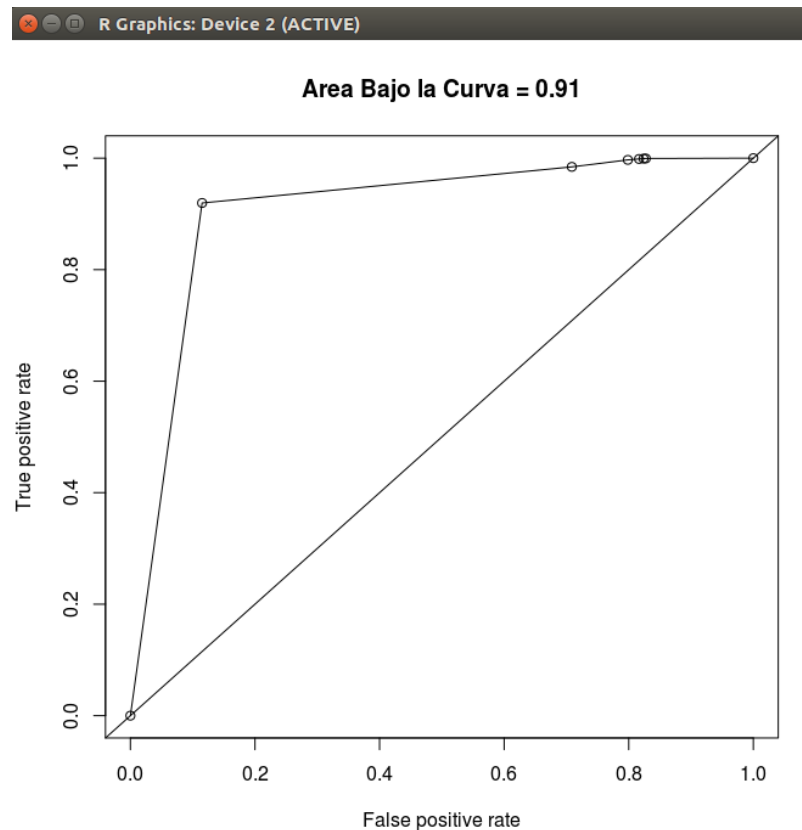


Ilustración 20. Curvas ROC para Rpart

Podemos ver que el valor obtenido es 0,91 por lo que se puede afirmar que es bueno.

Resulta ser un modelo bueno ya que ha arrojado resultados bastante aproximados .

- Random Forest

A continuación se explican los resultados de aplicar random forest el cual esta formado por múltiples árboles de decisión individuales.

Se ha usado este algoritmo puesto que se ha visto que devuelve resultados bastante buenos basados en la creación de un gran número de arboles de decisión con una selección aleatoria de un pequeño número de variables para cada uno. Usa el muestreo para cada selección de elementos con re emplazamiento.

Nos crea clasificadores individuales los cuales se caracterizan por estar bastante adaptados a los datos.



Como desventaja de este método puede encontrarse el posible sobreentrenamiento pero finalmente se ha de notar que este es compensado ya que los múltiples árboles son sobreentrenados de formas distintas por lo que sus errores al final, como se ha mencionado anteriormente, son compensados.

```
> rf.model_train <- randomForest(as.factor(entrenamiento$loans_status) ~.,
ntree = 1000,
type="classification",
importance=TRUE,
na.action=na.omit)
```

#Visualización random forest

```
> varImpPlot(rf.model_train)
> getTree(rf.model_train, 1)
```

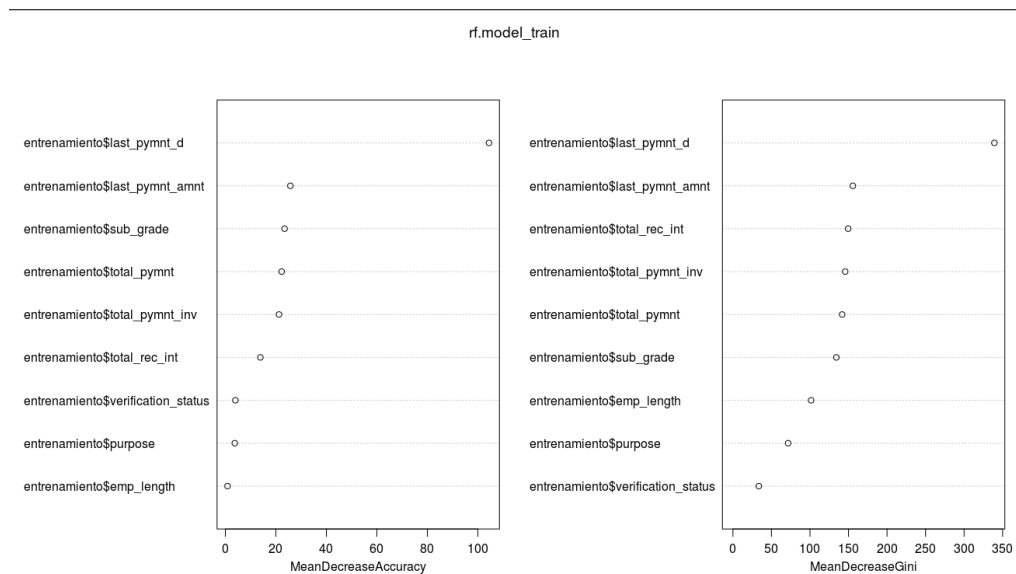


Ilustración 21. Medidas de importancia de cada uno de los atributos

Como se puede apreciar en la imagen anterior, volvemos a obtener de nuevo las valores más importantes donde vuelven a coincidir con los resultados obtenidos anteriormente.

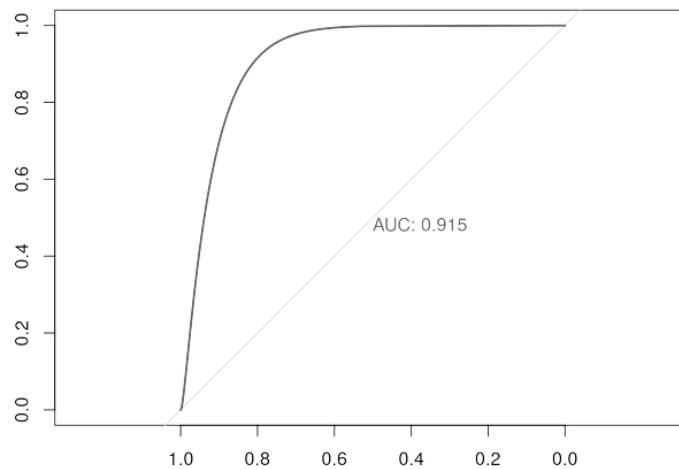


Ilustración 22. Curvas Roc con smooth

El valor obtenido es del 0,915 por lo que se podría decir que es un poco mejor que el anterior.

```
OOB estimate of error rate: 1.18%
Confusion matrix:
      Not_paid Paid class.error
Not_paid   150   499 0.768875193
Paid       96 49765 0.001925352
```

La puntuación del que se ha obtenido ha sido la siguiente la anteriormente vista con un error del 1,18 %.

En no pagados es de un 0,76 % mientras que en pagados es de un 0,0019 %.

- SVM

Las máquinas de vectores de soporte (SVM) nos van a permitir llevar a cabo la clasificación por medio de la partición en subespacios de varias variables.

La idea fundamental es crear muchos hiperplanos los cuales van a dividir las observaciones en grupos. Para ello se ha usado el paquete `e1071`.

Se a ejecutado el modelo SVM de la siguiente forma:

```
>modelo_svm=svm(entrenamiento$loans_status~.,data=entrenamiento,method="
C-classification", kernel="radial",cost=10,gamma=1)
```

A continuación se ha hecho la predicción de los restantes con:

```
> svm_pred <-predict(svm_model, validacion)
```

Y ahora hacemos la tabla de confusión para que aparezca la variable clase usamos `with`:

```
>(confu <- with(validacion, table(svm_pred, entrenamiento$loans_status))))
```

svpred		Not_paid	Paid
Not_paid		0	0
Paid		651	50364

Finalmente vemos el porcentaje de correctamente clasificados:

```
>(correctamente <-sum(diag(confu))/nrow(validacion)*100)
```

```
>svm_resul = cbind(entrenamiento, predic=svm_predic)
```

```
>table(svm_resul$loans_status, svm_resul$predict.modelo_svm..entrenamiento.)
```

Finalmente vemos la matriz:

	Not_paid	Paid
Not_paid	0	651
Paid	0	50364

Ilustración 23. Matriz

Ahora vemos las curvas ROC:

```
>roc_obj <- roc(entrenamiento$loans_status, as.numeric(svpred))
```

```
>auc(roc_obj)
```

El cual da como resultado 0.5 siendo por lo tanto peor que los métodos anteriores.

- Lineal Regression

```
>lin.reg <- lm(as.numeric(entrenamiento$loans_status) ~., data = entrenamiento)
```

```
>summary(lin.reg)
```

```
> summary(lin.reg)

Call:
lm(formula = as.numeric(entrenamiento$loans_status) ~ entrenamiento$purpose +
  entrenamiento$sub_grade + entrenamiento$emp_length + entrenamiento$total_pymnt_inv +
  entrenamiento$verification_status + entrenamiento$total_pymnt +
  entrenamiento$last_pymnt_amnt + entrenamiento$total_rec_int +
  entrenamiento$last_pymnt_d, data = entrenamiento)

Residuals:
    Min       1Q   Median       3Q      Max
-1.01108 -0.00341  0.00116  0.00803  0.27811

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)      1.019e+00  1.073e-02  95.048  < 2e-16 ***
entrenamiento$purposecredit_card    5.684e-03  4.160e-03   1.366  0.171836
entrenamiento$purposedebt_consolidation  3.894e-03  4.101e-03   0.949  0.342392
entrenamiento$purposeeducational    2.050e-02  9.654e-02   0.212  0.831820
entrenamiento$purposehome_improvement -2.713e-03  4.342e-03  -0.625  0.532064
entrenamiento$purposehouse    1.031e-03  5.769e-03   0.179  0.858196
entrenamiento$purposemajor_purchase -7.444e-04  4.815e-03  -0.155  0.877154
entrenamiento$purposemedical    5.737e-03  5.295e-03   1.083  0.278597
entrenamiento$purposemoving    -2.721e-04  6.436e-03  -0.042  0.966283
entrenamiento$purposeother    3.987e-03  4.304e-03   0.926  0.354290
entrenamiento$purposerenewable_energy -3.359e-02  1.660e-02  -2.024  0.042946 *
entrenamiento$purposesmall_business -4.604e-03  5.447e-03  -0.845  0.398037
entrenamiento$purposevacation    3.769e-03  6.327e-03   0.596  0.551382
entrenamiento$sub_gradeA2    -1.468e-03  3.502e-03  -0.419  0.675104
entrenamiento$sub_gradeA3    -2.369e-03  3.302e-03  -0.718  0.473051
entrenamiento$sub_gradeA4    -5.968e-03  3.298e-03  -1.810  0.070362 .
entrenamiento$sub_gradeA5    -3.476e-03  3.269e-03  -1.063  0.287643
entrenamiento$sub_gradeB1    -5.978e-03  3.171e-03  -1.885  0.059426 .
entrenamiento$sub_gradeB2    -3.787e-03  3.154e-03  -1.201  0.229913
entrenamiento$sub_gradeB3    -3.234e-03  3.143e-03  -1.029  0.303633
entrenamiento$sub_gradeB4    -7.080e-03  3.330e-03  -2.126  0.033504 *
entrenamiento$sub_gradeB5    -7.642e-03  3.140e-03  -2.434  0.014956 *
entrenamiento$sub_gradeC1    -9.874e-03  3.196e-03  -3.090  0.002006 **
```

```

entrenamiento$sub_gradeF1      -2.361e-02  7.199e-03  -3.279  0.001042 **
entrenamiento$sub_gradeF2      -3.666e-02  1.136e-02  -3.228  0.001245 **
entrenamiento$sub_gradeF3      -3.024e-02  1.063e-02  -2.844  0.004460 **
entrenamiento$sub_gradeF4      -5.032e-02  1.152e-02  -4.370  1.24e-05 ***
entrenamiento$sub_gradeF5      -5.123e-02  1.052e-02  -4.872  1.11e-06 ***
entrenamiento$sub_gradeG1      -5.111e-02  9.315e-03  -5.487  4.11e-08 ***
entrenamiento$sub_gradeG2      -5.281e-02  2.303e-02  -2.293  0.021844 *
entrenamiento$sub_gradeG3      -5.735e-02  1.965e-02  -2.919  0.003518 **
entrenamiento$sub_gradeG4      -1.128e-01  1.855e-02  -6.079  1.22e-09 ***
entrenamiento$sub_gradeG5      -1.020e-01  2.089e-02  -4.886  1.03e-06 ***
entrenamiento$emp_length< 1 year -1.709e-03  1.459e-03  -1.171  0.241427
entrenamiento$emp_length1 year  1.632e-03  1.770e-03  0.922  0.356699
entrenamiento$emp_length2 years  9.103e-04  1.546e-03  0.589  0.555952
entrenamiento$emp_length3 years  1.480e-04  1.645e-03  0.090  0.928323
entrenamiento$emp_length4 years -5.332e-04  1.788e-03  -0.298  0.765567
entrenamiento$emp_length5 years -1.517e-03  1.823e-03  -0.832  0.405385
entrenamiento$emp_length6 years -6.360e-04  2.090e-03  -0.304  0.760823
entrenamiento$emp_length7 years -2.089e-03  2.221e-03  -0.940  0.347013
entrenamiento$emp_length8 years -8.447e-04  2.580e-03  -0.327  0.743326
entrenamiento$emp_length9 years  1.972e-03  2.470e-03  0.799  0.424474
entrenamiento$emp_lengthn/a     1.132e-03  4.823e-02  0.023  0.981282
entrenamiento$total_pymnt_inv   6.032e-06  1.176e-04  0.051  0.959087
entrenamiento$verification_statusSource Verified -3.965e-03  9.841e-04  -4.030  5.59e-05 ***
entrenamiento$verification_statusVerified -5.031e-03  1.205e-03  -4.175  2.99e-05 ***
entrenamiento$total_pymnt      -4.549e-06  1.176e-04  -0.039  0.969134
entrenamiento$last_pymnt_amnt   -6.295e-06  9.453e-07  -6.659  2.79e-11 ***
entrenamiento$total_rec_int     8.651e-06  1.622e-06  5.333  9.71e-08 ***
entrenamiento$last_pymnt_dFeb-2018 9.849e-01  9.542e-03  103.221 < 2e-16 ***
entrenamiento$last_pymnt_dJan-2018 8.983e-01  9.630e-03  93.279 < 2e-16 ***
entrenamiento$last_pymnt_dNov-2017 1.239e-03  1.954e-02  0.063  0.949451
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.09641 on 50948 degrees of freedom
Multiple R-squared:  0.2632,    Adjusted R-squared:  0.2623 
F-statistic: 275.8 on 66 and 50948 DF,  p-value: < 2.2e-16

```

Ilustración 24. Regresión lineal

Aplicamos nuestro modelo sobre validación y evaluamos el accuracy.

```

> test.pred.lin <- exp(predict(lin.reg,test))-1

> RMSE.base
[1] 0.1117317
> RMSE.lin.reg
[1] 4.329084
> MAE.base
[1] 0.02508209
> MAE.lin.reg
[1] 4.318485

```

Ilustración 25. Accuracy

A continuacion vemos la grafica en relacion de loans\_status y purpose:

```
> abline(lin.reg)
```

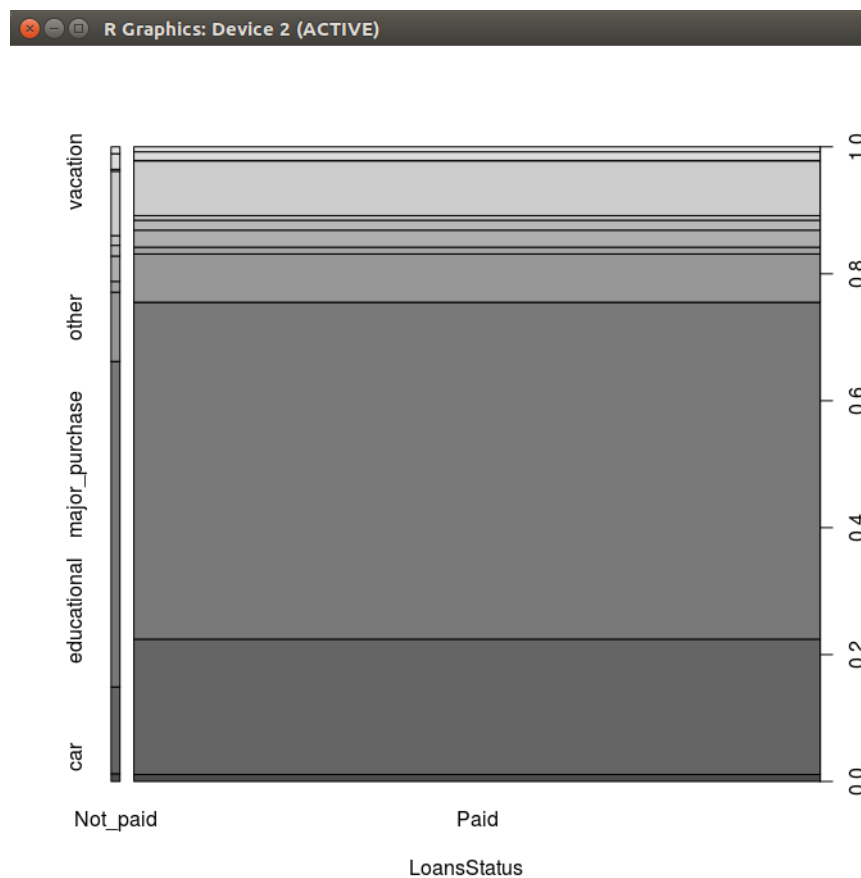


Ilustración 26. Relacion loans status y purpose

Ahora vemos los residuos y los valores ajustados:

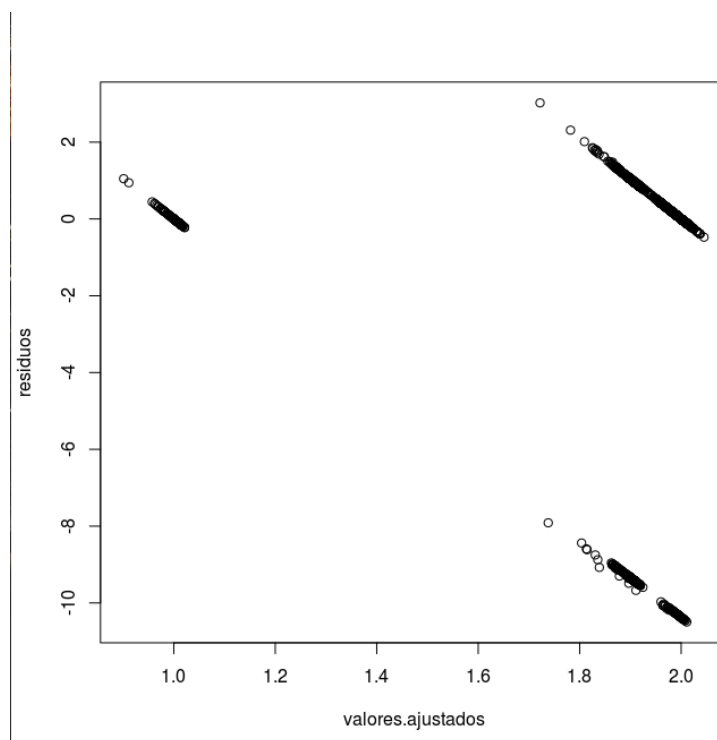


Ilustración 27. Residuos/valores ajustados

En ellos se observan tres patrones especiales.

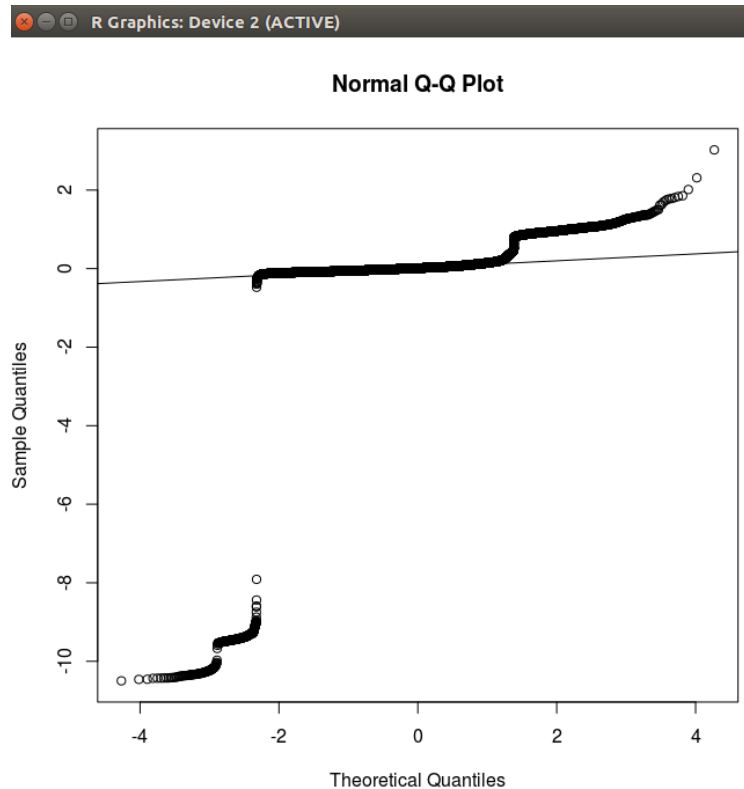


Ilustración 29. Normalidad

En ella no se ve un buen alineamiento ni una muy buena normalidad.

Como se ve en las imágenes anteriores el R-squared es de 0,26 lo que implica que el 26% de la varianza in nuestra variable dependiente puede ser explicada por nuestro modelo. Por lo que implica que no es un modelo muy bueno, ademas si vemos los valores de RMSE (root mean squared error) y de MAE (mean absolute error) se ha modificado bastante si los comparamos con el primer modelo base.

- Regresión logística

A continuación tenemos otro método que se ha estado probando pero el cual debido a la falta de tiempo no se ha podido terminar de realizar.

```
>glm.fit = glm( entrenamiento$loans_status~. ,data = entrenamiento , family = binomial )
```

```
>print(summary(glm.fit))
```

```
> print(contrasts(entrenamiento$loans_status))
```

```
      Paid
Not_paid  0
Paid      1
```

Ahora calculamos las probabilidades de ser Pagado o impagado siendo 1 pago y 0 impago como se ha visto en el resultado anterior.

```
> print(glm.probs[1:10])
```

```
      1      2      3      4      6      7      8      9
0.9986156 0.9997009 0.9951788 0.9986150 0.9936534 0.9821765 0.9998149 0.9962351
      10      11
0.9988011 0.9982599
```

Se han calculado las posibilidades de ser cada caso clasificado como pagado o no y se he visto como en el paso anterior que las primeras tienen mas probabilidad de ser clasificadas como pagado que como no pagado, para ello se ha lanzado lo siguiente:

```
>glm.probs <- predict ( glm.fit , type ="response")
```

```
>glm.pred <- rep ("Not_Paid" ,1250)
>glm.pred [ glm.probs >.5] <- "Paid"
```

Finalmente se ha hecho la matriz de confusión y se ha intentado mostrar el porcentaje de casos clasificados correctamente.



## 4. Conclusiones

Inicialmente el análisis del dataset ha sido una tarea laboriosa ya que debido a su tamaño y a la poca intuitividad de sus valores no se podía ver a primera vista las variables mas diferenciadoras. A ello se le ha sumado ademas el gran tiempo que ha tomado la limpieza en si del dataset la cual ha tomado la mayor cantidad del tiempo ya que se han hecho muchas pruebas para quedarse finalmente con las variables mas importantes.

Se ha de sumar ademas los numerosos problemas obtenidos para la obtención de los modelos ya que la gran mayoría han tomado bastantes horas en su ejecución lo que impedía que se tuviese mucho margen para testear las distintas posibilidades como por ejemplo en el caso de que se ha intentado aplicar Cforest pero debido a que tomaba mas de 5 horas y finalmente la maquina se quedaba bloqueada e incluso llegaba apagarse, no se ha podido probar correctamente.

Por lo que este clasificador y las regresiones lineal y logística quedarían por ejemplo como trabajo futuro.

Finalmente, se ha de destacar que el modelo que ha arrojado mejores resultados ha sido el de r-part y random forest.

El problema del dataset ha sido principalmente el gran desequilibrio entre todos los valores.

## Bibliografía

- <http://trevorstephens.com/kaggle-titanic-tutorial/getting-started-with-r/>
- <https://www.statmethods.net/r-tutorial/index.html>
- <https://www.r-bloggers.com/calculating-auc-the-area-under-a-roc-curve/>
- <https://www.r-bloggers.com/how-to-perform-a-logistic-regression-in-r/>
- <https://www.r-bloggers.com/how-to-learn-r-2/>
- <https://rpro.wikispaces.com/Modelos%20para%20explicar%2C%20ordenar%20y%20clasificar>
- <https://www.r-bloggers.com/machine-learning-using-support-vector-machines/>