Mantenimiento Preventivo de Máquinas

Machine Learning Predictivo

15/2/2022

---

1 ID Petición

- Número 3: Predicción del fallo en una máquina para mantenimiento preventivo.

2 Contexto

- El equipo de dirección quiere tener un plan de mantenimiento preventivo de la máquinaria con la que cuenta la empresa.
- Realizar el mantenimiento preventivo de la maquinaria evita averías en la maquína, además de cortes en la producción y disminución en el coste de mano de obra.
- El departamento de mantenimiento quiere un plan concreto de mantenimiento preventivo de la maquinaria de la empresa, para organización de las tareas del personal del departamento.

3 Requirimientos del proyecto

- Realización del plan de mantenimiento preventivo.
- Optimizar las labores del personal de mantenimiento.

---

4 Detalle del proceso

4.1 - Preparación del entorno

- 4.1.1 - Cargamos las librerías que vamos a utilizar

```
paquetes <- c('dplyr',
              'skimr',
              'lubridate',
              'tidyr',
              'ggplot2',
              'ROCR')

instalados <- paquetes %in% installed.packages()

if(sum(instalados == FALSE) > 0) {
  install.packages(paquetes[!instalados])
}
lapply(paquetes,require,character.only = TRUE)
```

```
## [[1]]
## [1] TRUE
##
## [[2]]
## [1] TRUE
##
## [[3]]
## [1] TRUE
##
## [[4]]
## [1] TRUE
##
## [[5]]
## [1] TRUE
##
## [[6]]
## [1] TRUE
```

Parámetros - Desactivamos la notación científica:

```
options(scipen=999)
```

- 4.1.2 - Cargamos los datos

Procedentes de la base de datos histórica de la compañía, archivo '.csv'.

```
df <- read.csv('MntoMaquina.csv')
```

4.2 - Análisis exploratorio

- 4.2.1 - Análisis exploratorio general y tipo de datos

Vemos las variables con las que trabajamos y el tipo de dato de cada una.

```
glimpse(df)
```

```
## Rows: 7,027
## Columns: 28
## $ ï..Date                      <chr> "2016-01-01 01:00:00", "2016-01-01 02:00:~
## $ Temperature                  <int> 68, 64, 63, 65, 67, 65, 63, 61, 62, 62, 6~
## $ Humidity                     <int> 77, 76, 80, 81, 76, 80, 80, 83, 81, 76, 8~
## $ Operator                     <chr> "Operator1", "Operator1", "Operator1", "O~
## $ Measure1                     <int> 1180, 1406, 550, 1928, 1021, 1731, 415, 5~
## $ Measure2                     <int> 1, 1, 1, 1, 2, 2, 0, 2, 3, 0, 1, 3, 0, 3,~
## $ Measure3                     <int> 1, 1, 1, 2, 1, 0, 0, 2, 1, 0, 1, 0, 2, 2,~
## $ Measure4                     <int> 1915, 511, 1754, 1326, 185, 1424, 1008, 6~
## $ Measure5                     <int> 1194, 1577, 1834, 1082, 170, 1176, 1086, ~
## $ Measure6                     <int> 637, 1121, 1413, 233, 952, 1223, 1759, 17~
## $ Measure7                     <int> 1093, 1948, 1151, 1441, 1183, 621, 1946, ~
## $ Measure8                     <int> 524, 1882, 945, 1736, 1329, 647, 1814, 64~
## $ Measure9                     <int> 919, 1301, 1312, 1033, 427, 369, 1754, 31~
## $ Measure10                    <int> 245, 273, 1494, 1549, 1638, 239, 1442, 93~
## $ Measure11                    <int> 403, 1927, 1755, 802, 850, 1196, 341, 189~
## $ Measure12                    <int> 723, 1123, 1434, 1819, 379, 1944, 1097, 1~
## $ Measure13                    <int> 1446, 717, 502, 1616, 1529, 1583, 1819, 1~
## $ Measure14                    <int> 719, 1518, 1336, 1507, 755, 1630, 472, 15~
## $ Measure15                    <int> 748, 1689, 711, 507, 844, 237, 491, 1102,~
## $ Hours.Since.Previous.Failure <int> 91, 92, 93, 94, 97, 98, 99, 100, 101, 102~
## $ Failure                      <chr> "No", "No", "No", "No", "No", "No", "No",~
## $ Date.year                    <int> 2016, 2016, 2016, 2016, 2016, 2016, 2016,~
## $ Date.month                   <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,~
## $ Date.day.of.month            <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,~
## $ Date.day.of.week             <int> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,~
## $ Date.hour                    <int> 1, 2, 3, 4, 7, 8, 9, 10, 11, 12, 13, 14, ~
## $ Date.minute                  <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ Date.second                  <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
```

Tenemos un conjunto de datos de 28 columnas y 7027 filas.

Esta función nos ofrece unos pequeños gráficos para ver el perfil de las variables, así como los datos faltantes, la media, los máximos y mínimos y los percentiles por cada variable.

```
skim(df)
```

Data summary

Name	df
Number of rows	7027
Number of columns	28
_____	
Column type frequency:	
character	3
numeric	25
_____	
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
ï..Date	0	1	19	19	0	7027	0
Operator	0	1	9	9	0	8	0

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
Failure	0	1	2	3	0	2	0

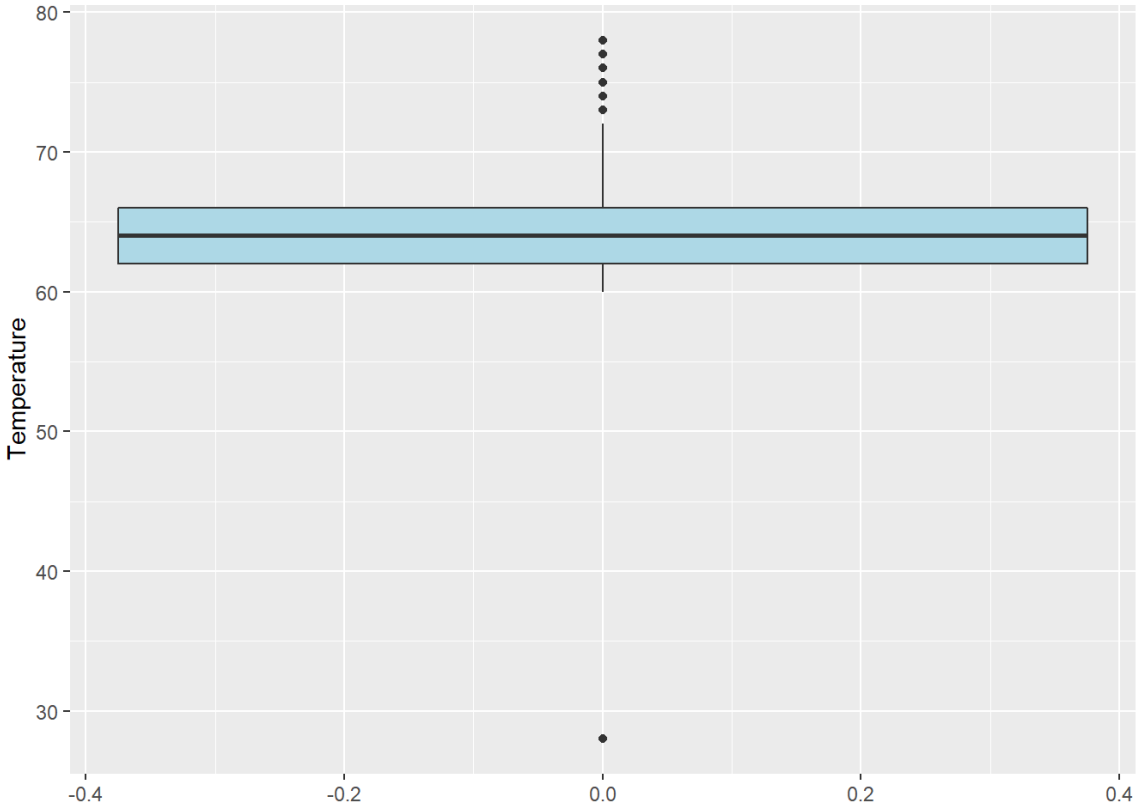
Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
Temperature	0	1	64.03	2.70	28	62.0	64	66.0	78	
Humidity	0	1	83.33	4.85	65	80.0	83	87.0	122	
Measure1	0	1	1090.76	536.00	155	631.0	1090	1555.0	2011	
Measure2	0	1	1.50	1.12	0	0.0	2	2.0	3	
Measure3	0	1	1.01	0.82	0	0.0	1	2.0	2	
Measure4	0	1	1077.52	537.16	155	617.0	1069	1541.0	2011	
Measure5	0	1	1067.81	531.94	155	596.0	1070	1531.0	2011	
Measure6	0	1	1075.19	534.40	155	622.0	1072	1538.0	2011	
Measure7	0	1	1089.31	539.90	155	618.5	1091	1563.0	2011	
Measure8	0	1	1074.99	538.53	155	606.5	1075	1534.0	2011	
Measure9	0	1	1080.66	531.36	155	629.5	1078	1525.0	2011	
Measure10	0	1	1078.82	537.29	155	617.0	1073	1541.5	2011	
Measure11	0	1	1090.72	535.17	155	627.0	1097	1552.5	2011	
Measure12	0	1	1088.06	531.60	155	631.0	1083	1547.5	2011	
Measure13	0	1	1076.29	534.08	155	605.0	1071	1540.0	2011	
Measure14	0	1	1085.28	538.22	155	611.0	1087	1558.5	2011	
Measure15	0	1	1085.99	537.12	155	620.0	1079	1554.0	2011	
Hours.Since.Previous.Failure	0	1	216.08	152.39	1	88.0	192	323.0	666	
Date.year	0	1	2016.00	0.00	2016	2016.0	2016	2016.0	2016	
Date.month	0	1	6.51	3.46	1	3.0	7	10.0	12	
Date.day.of.month	0	1	15.75	8.81	1	8.0	16	23.0	31	
Date.day.of.week	0	1	4.01	2.00	1	2.0	4	6.0	7	
Date.hour	0	1	11.47	6.91	0	5.0	11	17.0	23	
Date.minute	0	1	0.00	0.00	0	0.0	0	0.0	0	
Date.second	0	1	0.00	0.00	0	0.0	0	0.0	0	

De la observación de estos datos, podemos concluir: - No hay nulos - Measure2 y Measure3 parecen factores, en lugar de enteros. - Viendo el mínimo y el p25 de Temperature parece que tiene algunos datos atípicos.

Analizamos en mayor detalle la tempertura y los datos atípicos que hemos detectado.

```
ggplot(df,x=1) + geom_boxplot(aes(y=Temperature), fill='lightblue')
```



Conclusión: efectivamente vemos que hay varios valores atípicos por debajo del cuartil 25.

- 4.2.2 - Calidad de datos:

Hacemos las siguientes acciones sobre el dataframe original:

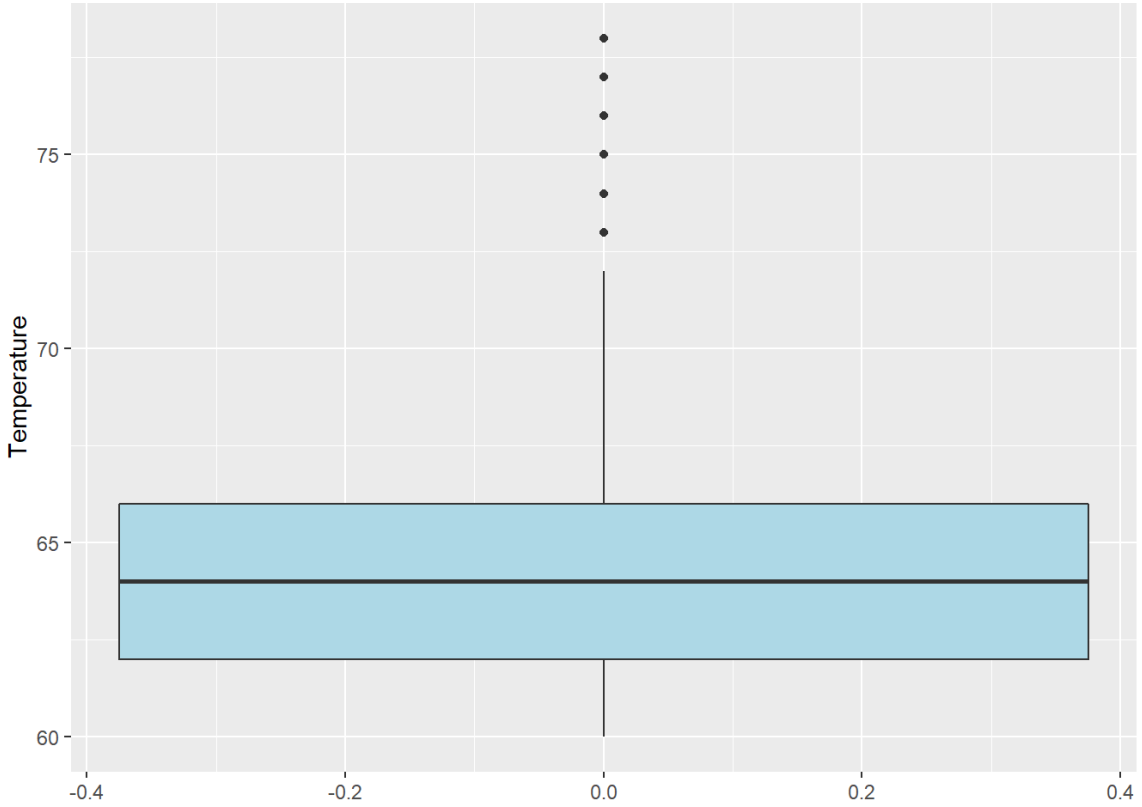
Pasamos a factor las variables Measure 2 y 3, además de la variable del Operator y la de Failure, que solo tiene valores ‘Yes’ y ‘No’.

También eliminamos los valores de temperatura que estén por debajo de 45 grados, los cuales hemos considerado valores atípicos.

```
df <- df %>%
  mutate(Measure2 = as.factor(Measure2),
         Measure3 = as.factor(Measure3),
         Operator = as.factor(Operator),
         Failure = as.factor(Failure)) %>%
  filter(Temperature > 45)
```

Comprobamos la variable temperatura con un gráfico.

```
ggplot(df,x=1) + geom_boxplot(aes(y=Temperature), fill='lightblue')
```



Comprobamos también que Measure 2 y 3 han pasado a tipo de datos: factor.

```
glimpse(df)
```

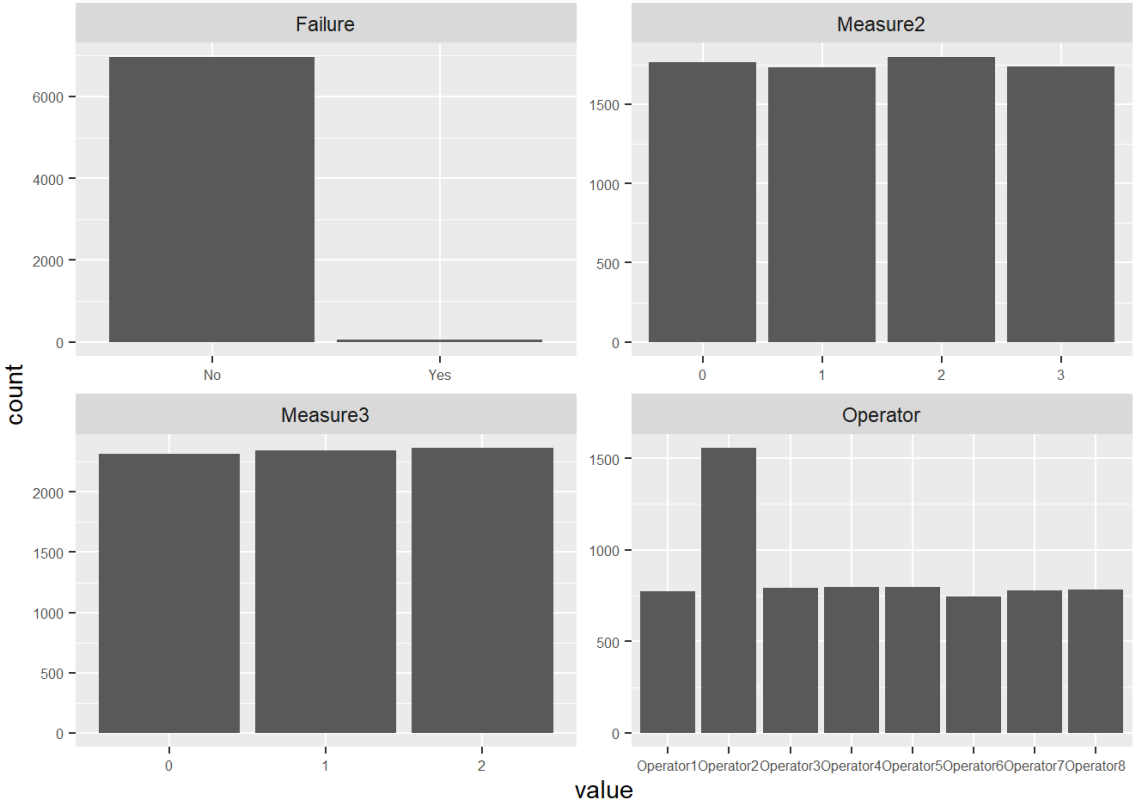
```
## Rows: 7,026
## Columns: 28
## $ ï..Date                     <chr> "2016-01-01 01:00:00", "2016-01-01 02:00:~
## $ Temperature                 <int> 68, 64, 63, 65, 67, 65, 63, 61, 62, 62, 6~
## $ Humidity                    <int> 77, 76, 80, 81, 76, 80, 80, 83, 81, 76, 8~
## $ Operator                    <fct> Operator1, Operator1, Operator1, Operator~
## $ Measure1                    <int> 1180, 1406, 550, 1928, 1021, 1731, 415, 5~
## $ Measure2                    <fct> 1, 1, 1, 1, 2, 2, 0, 2, 3, 0, 1, 3, 0, 3,~
## $ Measure3                    <fct> 1, 1, 1, 2, 1, 0, 0, 2, 1, 0, 1, 0, 2, 2,~
## $ Measure4                    <int> 1915, 511, 1754, 1326, 185, 1424, 1008, 6~
## $ Measure5                    <int> 1194, 1577, 1834, 1082, 170, 1176, 1086, ~
## $ Measure6                    <int> 637, 1121, 1413, 233, 952, 1223, 1759, 17~
## $ Measure7                    <int> 1093, 1948, 1151, 1441, 1183, 621, 1946, ~
## $ Measure8                    <int> 524, 1882, 945, 1736, 1329, 647, 1814, 64~
## $ Measure9                    <int> 919, 1301, 1312, 1033, 427, 369, 1754, 31~
## $ Measure10                   <int> 245, 273, 1494, 1549, 1638, 239, 1442, 93~
## $ Measure11                   <int> 403, 1927, 1755, 802, 850, 1196, 341, 189~
## $ Measure12                   <int> 723, 1123, 1434, 1819, 379, 1944, 1097, 1~
## $ Measure13                   <int> 1446, 717, 502, 1616, 1529, 1583, 1819, 1~
## $ Measure14                   <int> 719, 1518, 1336, 1507, 755, 1630, 472, 15~
## $ Measure15                   <int> 748, 1689, 711, 507, 844, 237, 491, 1102,~
## $ Hours.Since.Previous.Failure <int> 91, 92, 93, 94, 97, 98, 99, 100, 101, 102~
## $ Failure                     <fct> No, No, No, No, No, No, No, No, No, No, N~
## $ Date.year                   <int> 2016, 2016, 2016, 2016, 2016, 2016, 2016,~
## $ Date.month                  <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,~
## $ Date.day.of.month           <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,~
## $ Date.day.of.week            <int> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,~
## $ Date.hour                   <int> 1, 2, 3, 4, 7, 8, 9, 10, 11, 12, 13, 14, ~
## $ Date.minute                 <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ Date.second                 <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
```

El dataframe actualmente consta de 7026 filas, se ha eliminado la fila del valor atípico de la Temperatura.

- 4.2.3 - Calidad de datos: Análisis de atípicos

4.2.3.1 - Analizamos las que son de tipo factor

```
df %>%
  select_if(is.factor) %>%
  gather() %>%
  ggplot(aes(value)) + geom_bar() + facet_wrap(~key,scales='free') + theme(axis.text=element_text
(size=6))
```

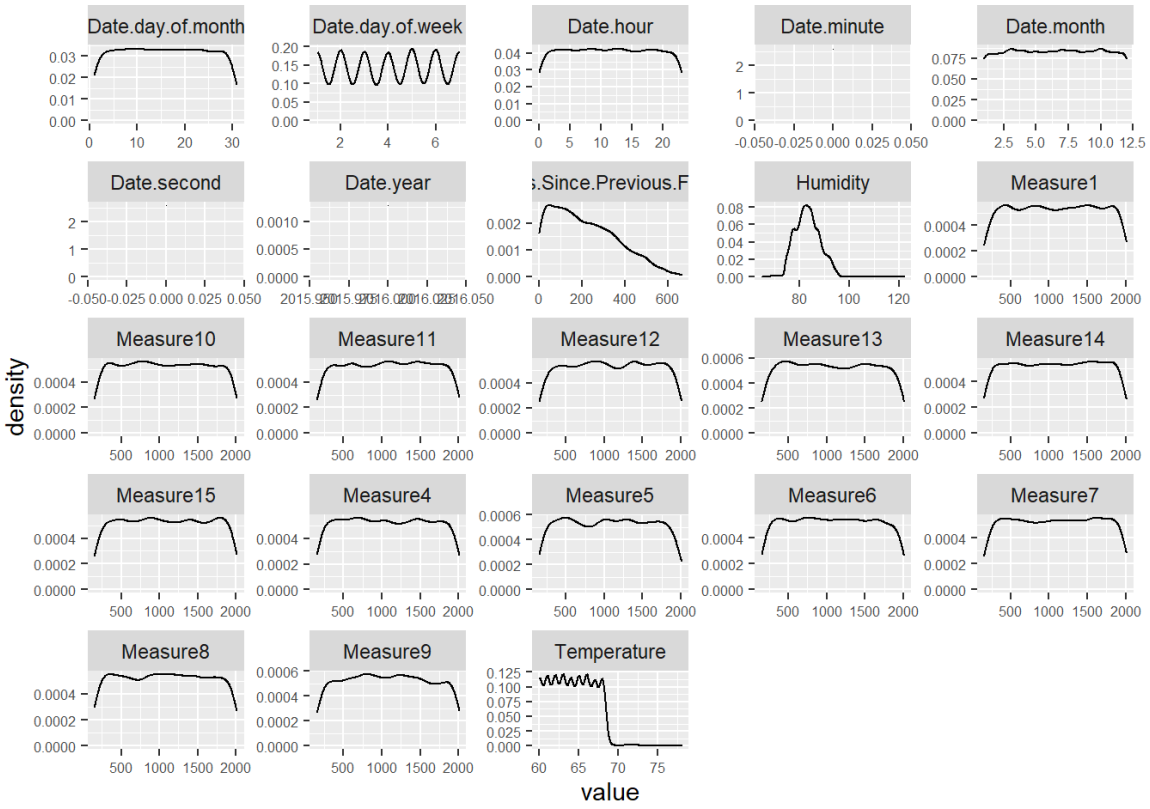


Obtenemos cuatro gráficos ya que son las variables tipo factor que tiene el conjunto de datos. El gráfico Failure está muy desequilibrado ya que el número de fallos es muy bajo respecto al número de no fallos. Después realizaremos operaciones para compensar este desequilibrio.

En los gráficos de Measure 2 y 3 vemos un reparto uniforme de las categorías que nos han salido. En el gráfico de operador, vemos que el número dos tiene mas interacciones que el resto, que son bastante similares.

4.2.3.2 - Analizamos las que son de tipo entero

```
df %>%
  select_if(is.integer) %>%
  gather() %>%
  ggplot(aes(value)) + geom_density() + facet_wrap(~key,scales='free') +
theme(axis.text=element_text(size=6))
```



El gráfico de Horas desde el último fallo nos indica que en las primeras 50 horas de trabajo, las máquinas no suelen fallar, pero cuando vamos aumentando las horas de trabajo, va disminuyendo el número de horas que pasa entre un fallo y el siguiente. El resto de variables presentan patrones coherentes con el sentido de negocio.

Vemos que las variables de fecha, día de la semana, año, mes, minuto y segundo, no nos ofrecen información relevante, así que las eliminamos de nuestro conjunto de datos.

```
df <- df %>%
  select(- c(ï..Date,Date.year,Date.month,Date.day.of.month,Date.day.of.week,Date.hour,Date.minut
e,Date.second))
glimpse(df)
```

```
## Rows: 7,026
## Columns: 20
## $ Temperature                 <int> 68, 64, 63, 65, 67, 65, 63, 61, 62, 62, 6~
## $ Humidity                    <int> 77, 76, 80, 81, 76, 80, 80, 83, 81, 76, 8~
## $ Operator                    <fct> Operator1, Operator1, Operator1, Operator~
## $ Measure1                    <int> 1180, 1406, 550, 1928, 1021, 1731, 415, 5~
## $ Measure2                    <fct> 1, 1, 1, 1, 2, 2, 0, 2, 3, 0, 1, 3, 0, 3,~
## $ Measure3                    <fct> 1, 1, 1, 2, 1, 0, 0, 2, 1, 0, 1, 0, 2, 2,~
## $ Measure4                    <int> 1915, 511, 1754, 1326, 185, 1424, 1008, 6~
## $ Measure5                    <int> 1194, 1577, 1834, 1082, 170, 1176, 1086, ~
## $ Measure6                    <int> 637, 1121, 1413, 233, 952, 1223, 1759, 17~
## $ Measure7                    <int> 1093, 1948, 1151, 1441, 1183, 621, 1946, ~
## $ Measure8                    <int> 524, 1882, 945, 1736, 1329, 647, 1814, 64~
## $ Measure9                    <int> 919, 1301, 1312, 1033, 427, 369, 1754, 31~
## $ Measure10                   <int> 245, 273, 1494, 1549, 1638, 239, 1442, 93~
## $ Measure11                   <int> 403, 1927, 1755, 802, 850, 1196, 341, 189~
## $ Measure12                   <int> 723, 1123, 1434, 1819, 379, 1944, 1097, 1~
## $ Measure13                   <int> 1446, 717, 502, 1616, 1529, 1583, 1819, 1~
## $ Measure14                   <int> 719, 1518, 1336, 1507, 755, 1630, 472, 15~
## $ Measure15                   <int> 748, 1689, 711, 507, 844, 237, 491, 1102,~
## $ Hours.Since.Previous.Failure <int> 91, 92, 93, 94, 97, 98, 99, 100, 101, 102~
## $ Failure                     <fct> No, No, No, No, No, No, No, No, No, No, N~
```

Ahora el data frame tiene 7023 filas de 20 variables.

- 4.2.4 - Calidad de datos: Análisis de correlación

Estudiamos la correlación, porque nos interesa que las variables no estén muy correlacionadas para que la modelización posterior sea mas efectiva.

```
df %>%
  select_if(is.integer) %>%
  cor() %>%
  round(digits = 2)
```

```
##                              Temperature Humidity Measure1 Measure4 Measure5
## Temperature                         1.00    -0.05     0.01    -0.02     0.01
## Humidity                           -0.05     1.00     0.00     0.01    -0.03
## Measure1                            0.01     0.00     1.00     0.01     0.03
## Measure4                           -0.02     0.01     0.01     1.00     0.00
## Measure5                            0.01    -0.03     0.03     0.00     1.00
## Measure6                            0.00    -0.01     0.01     0.02     0.00
## Measure7                           -0.01    -0.02     0.00     0.00    -0.01
## Measure8                            0.00     0.02     0.00     0.00    -0.01
## Measure9                           -0.02     0.00    -0.01     0.01    -0.01
## Measure10                          -0.02     0.00     0.01    -0.02    -0.01
## Measure11                           0.01     0.03     0.00     0.00     0.01
## Measure12                           0.00    -0.01     0.00     0.02     0.01
## Measure13                          -0.01    -0.01    -0.02    -0.01     0.00
## Measure14                           0.00     0.00     0.01    -0.01     0.01
## Measure15                          -0.01    -0.01     0.00     0.00     0.01
## Hours.Since.Previous.Failure       -0.01     0.00     0.00    -0.01     0.01
##                              Measure6 Measure7 Measure8 Measure9 Measure10
## Temperature                      0.00    -0.01     0.00    -0.02     -0.02
## Humidity                        -0.01    -0.02     0.02     0.00      0.00
## Measure1                         0.01     0.00     0.00    -0.01      0.01
## Measure4                         0.02     0.00     0.00     0.01     -0.02
## Measure5                         0.00    -0.01    -0.01    -0.01     -0.01
## Measure6                         1.00     0.01     0.00     0.00      0.01
## Measure7                         0.01     1.00     0.01     0.00     -0.02
## Measure8                         0.00     0.01     1.00     0.00     -0.01
## Measure9                         0.00     0.00     0.00     1.00     -0.01
## Measure10                        0.01    -0.02    -0.01    -0.01      1.00
## Measure11                       -0.01     0.01    -0.01     0.00      0.01
## Measure12                        0.02    -0.01    -0.01     0.01      0.01
## Measure13                       -0.01     0.00     0.00    -0.01      0.02
## Measure14                       -0.01     0.01    -0.02    -0.01      0.00
## Measure15                        0.00    -0.01     0.01     0.03     -0.02
## Hours.Since.Previous.Failure    -0.01     0.00     0.00    -0.01     -0.01
##                              Measure11 Measure12 Measure13 Measure14 Measure15
## Temperature                       0.01      0.00     -0.01      0.00     -0.01
## Humidity                          0.03     -0.01     -0.01      0.00     -0.01
## Measure1                          0.00      0.00     -0.02      0.01      0.00
## Measure4                          0.00      0.02     -0.01     -0.01      0.00
## Measure5                          0.01      0.01      0.00      0.01      0.01
## Measure6                         -0.01      0.02     -0.01     -0.01      0.00
## Measure7                          0.01     -0.01      0.00      0.01     -0.01
## Measure8                         -0.01     -0.01      0.00     -0.02      0.01
## Measure9                          0.00      0.01     -0.01     -0.01      0.03
## Measure10                         0.01      0.01      0.02      0.00     -0.02
## Measure11                         1.00     -0.01     -0.01      0.00      0.01
## Measure12                        -0.01      1.00      0.01      0.00      0.01
## Measure13                        -0.01      0.01      1.00      0.01      0.01
## Measure14                         0.00      0.00      0.01      1.00      0.02
## Measure15                         0.01      0.01      0.01      0.02      1.00
## Hours.Since.Previous.Failure      0.00     -0.02      0.00      0.00     -0.01
##                              Hours.Since.Previous.Failure
## Temperature                                       -0.01
## Humidity                                           0.00
## Measure1                                           0.00
## Measure4                                          -0.01
## Measure5                                           0.01
## Measure6                                          -0.01
## Measure7                                           0.00
## Measure8                                           0.00
## Measure9                                          -0.01
## Measure10                                         -0.01
## Measure11                                          0.00
```

```
## Measure12                                    -0.02
## Measure13                                     0.00
## Measure14                                     0.00
## Measure15                                    -0.01
## Hours.Since.Previous.Failure                  1.00
```

Vemos que los valores obtenidos son próximos a cero lo cual indica que las variables no están muy correlaccionadas entre si.

- 4.2.5 - Calidad de datos: Desbalanceo

Vamos a comprobar como está de desbalanceada la variable 'Failure'.

```
table(df$Failure)
```

```
##
##   No  Yes
## 6961   65
```

Efectivamente vemos que la variable'Failure', que vamos a utilizar como target está muy desbalanceada y tenemos que corregirlo.

4.3 - Trasformación de datos

- 4.3.1 - Corrección del desbalanceo

Para corregir el desbalanceo vamos a utilizar la técnica del inframuestreo, con la que vamos a comprobar la penetración exacta de la target.

```
65/nrow(df) * 100
```

```
## [1] 0.9251352
```

Tenemos 65 'Yes' que sobre el total de los casos supone un 0.92%

Para obtener casi un 10% de 'Yes', necesitarímos incrementar la proporción aproximadamente unas 10 veces. Lo que vamos a hacer es reducir los 'No' para obtener esa proporción del 10 % de 'Yes'.

Generamos un nuevo dataframe con los 'No' y otro con los 'Yes'. El de los 'No' lo reducimos en tamaño a un 8%.

```
df_nos <- df %>%
  filter(Failure == 'No') %>%
  sample_frac(size = 0.08)
dim(df_nos)
```

```
## [1] 557  20
```

```
df_sis <- df %>% filter(Failure == 'Yes')
dim(df_sis)
```

```
## [1] 65 20
```

Generamos de nuevo un df reducido con los dos nuevos creados. Y comprobamos:

```
df_red <- rbind(df_nos,df_sis)
count(df_red,Failure)
```

```
##   Failure   n
## 1      No 557
## 2     Yes  65
```

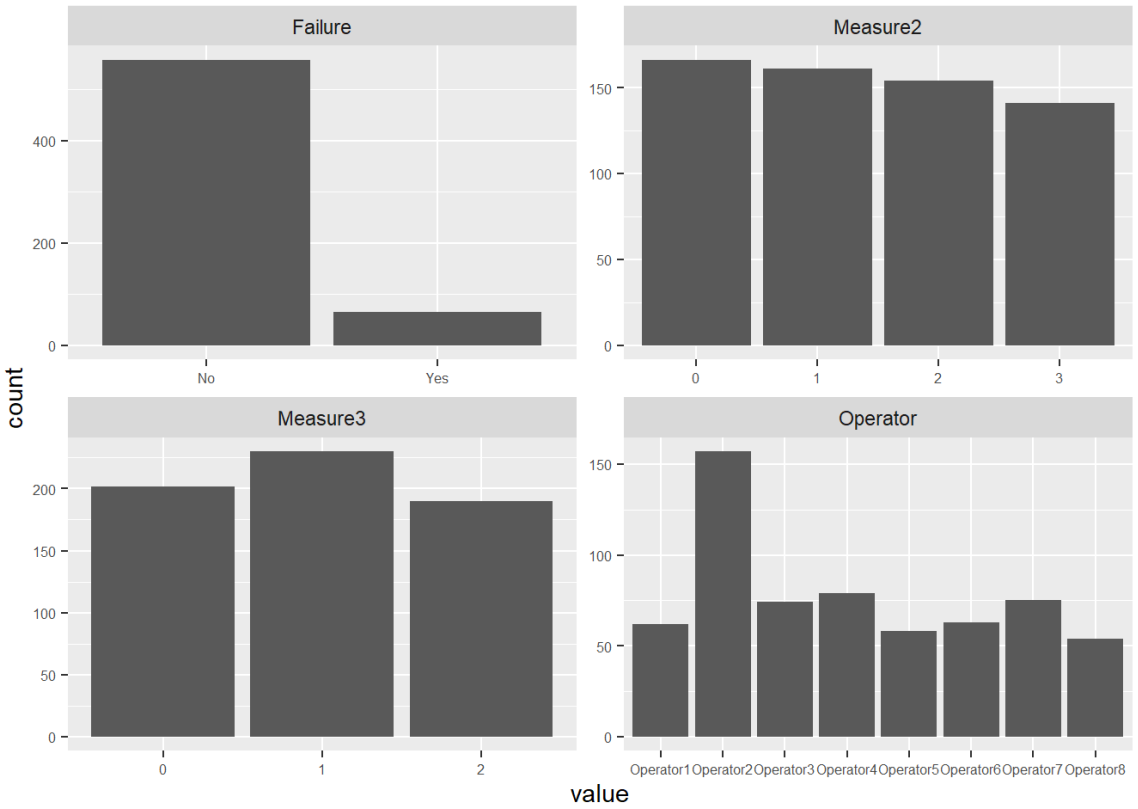
```
65/nrow(df_red) * 100
```

```
## [1] 10.45016
```

De esta manera, hemos obtenido un nuevo dataframe, en el que el 10 % de los datos son 'Yes', eso supone que la variable target tiene un 10% de penetración.

Visualizamos de nuevo los gráficos de las variables tipo factor, para comprobar como hemos aumentado la proporción de 'Yes', respecto al número de 'No'.

```
df_red %>%
  select_if(is.factor) %>%
  gather() %>%
  ggplot(aes(value)) + geom_bar() + facet_wrap(~key,scales='free') + theme(axis.text=element_text
(size=6))
```



4.4 - Modelización

- 4.4.1 - Preparacion de las funciones que vamos a necesitar

Función para crear una matriz de confusión.

```
confusion<-function(real,scoring,umbral){
  conf<-table(real,scoring>=umbral)
  if(ncol(conf)==2) return(conf) else return(NULL)
}
```

Funcion para calcular las métricas de los modelos: acierto, precisión, cobertura y F1.

```
metricas<-function(matriz_conf){
  acierto <- (matriz_conf[1,1] + matriz_conf[2,2]) / sum(matriz_conf) *100
  precision <- matriz_conf[2,2] / (matriz_conf[2,2] + matriz_conf[1,2]) *100
  cobertura <- matriz_conf[2,2] / (matriz_conf[2,2] + matriz_conf[2,1]) *100
  F1 <- 2*precision*cobertura/(precision+cobertura)
  salida<-c(acierto,precision,cobertura,F1)
  return(salida)
}
```

Función para probar distintos umbrales y ver el efecto sobre precisión y cobertura.

```
umbrales<-function(real,scoring){
  umbrales<-data.frame(umbral=rep(0,times=19),acierto=rep(0,times=19),precision=rep(0,times=19),c
obertura=rep(0,times=19),F1=rep(0,times=19))
  cont <- 1
  for (cada in seq(0.05,0.95,by = 0.05)){
    datos<-metricas(confusion(real,scoring,cada))
    registro<-c(cada,datos)
    umbrales[cont,]<-registro
    cont <- cont + 1
  }
  return(umbrales)
}
```

Funciones que calculan la curva ROC y el AUC.

```
roc<-function(prediction){
  r<-performance(prediction,'tpr','fpr')
  plot(r, col='darkgreen')
}

auc<-function(prediction){
  a<-performance(prediction,'auc')
  return(a@y.values[[1]])
}
```

- 4.4.2 - Creamos las particiones de training (70%) y test (30%)

Generamos una variable aleatoria con una distribución 70-30

```
df_red$random<-sample(0:1,size = nrow(df_red),replace = T,prob = c(0.3,0.7))
```

Creamos los dos dataframes. Y eliminamos la random generada.

```
train<-filter(df_red,random==1)
test<-filter(df_red,random==0)

df_red$random <- NULL
```

- 4.4.3 - Elección del modelo

Vamos a realizar la modelización con Regresión Logística y posteriormente evaluar el modelo.

4.4.3.1 - Identificación de las variables

Concretamos que la variable Target va a ser la variable ‘Failure’.

```
Target <- 'Failure'
```

Eliminamos la variable Target original. Ya que las variables predictoras para el modelo son todas las demas excepto ‘Failure’. Creamos la formula que vamos a pasar al modelo.

```
indep <- names(df_red)[-20]
formula <- reformulate(indep,Target)
```

Vamos a modelizar con una regresión logística.

```
formula_rl <- formula
rl <- glm(formula_rl,train,family=binomial(link='logit'))
summary(rl)
```

```
##
## Call:
## glm(formula = formula_rl, family = binomial(link = "logit"),
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.0754  -0.2113  -0.0802  -0.0273   3.2950
##
## Coefficients:
##                                 Estimate  Std. Error z value   Pr(>|z|)
## (Intercept)                  -1.65500166 10.66129617  -0.155     0.8766
## Temperature                   0.45776559  0.10874888   4.209 0.00002561 ***
## Humidity                     -0.38132831  0.08002382  -4.765 0.00000189 ***
## OperatorOperator2            -1.28982333  1.05163956  -1.226     0.2200
## OperatorOperator3            -1.04105309  1.29085665  -0.806     0.4200
## OperatorOperator4             0.28651472  1.09596092   0.261     0.7938
## OperatorOperator5            -1.06908515  1.34400437  -0.795     0.4264
## OperatorOperator6            -1.14343996  1.30884605  -0.874     0.3823
## OperatorOperator7             0.37445371  0.96097517   0.390     0.6968
## OperatorOperator8             0.44926471  1.24023446   0.362     0.7172
## Measure1                     -0.00044593  0.00059647  -0.748     0.4547
## Measure21                     0.01165833  0.89698619   0.013     0.9896
## Measure22                     0.93778318  0.86546948   1.084     0.2786
## Measure23                     0.08141948  0.95544815   0.085     0.9321
## Measure31                    -0.16536505  0.70716447  -0.234     0.8151
## Measure32                     0.78402960  0.70707148   1.109     0.2675
## Measure4                      0.00010368  0.00053080   0.195     0.8451
## Measure5                     -0.00011263  0.00058972  -0.191     0.8485
## Measure6                      0.00023111  0.00052184   0.443     0.6579
## Measure7                     -0.00015762  0.00054574  -0.289     0.7727
## Measure8                      0.00007369  0.00057913   0.127     0.8987
## Measure9                     -0.00050586  0.00054987  -0.920     0.3576
## Measure10                     0.00127946  0.00057381   2.230     0.0258 *
## Measure11                     0.00018095  0.00058449   0.310     0.7569
## Measure12                    -0.00071859  0.00061609  -1.166     0.2435
## Measure13                    -0.00001023  0.00058351  -0.018     0.9860
## Measure14                     0.00003232  0.00054840   0.059     0.9530
## Measure15                    -0.00036101  0.00055474  -0.651     0.5152
## Hours.Since.Previous.Failure -0.00242043  0.00200183  -1.209     0.2266
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 299.71  on 442  degrees of freedom
## Residual deviance: 102.54  on 414  degrees of freedom
## AIC: 160.54
##
## Number of Fisher Scoring iterations: 8
```

Como variables predictivas, con al menos el 90%, sólo resultan ‘Temperature’, ‘Humidity’, ‘Measure10’ ‘Hours.Since.Previous.Failure’, así que las seleccionamos como finales.

```
indep_fin <- c('Temperature','Humidity','Hours.Since.Previous.Failure')
formula_rl <- reformulate(indep_fin,Target)
```

Y volvemos a modelizar

```
rl <- glm(formula_rl,train,family=binomial(link='logit'))
summary(rl)
```

```
##
## Call:
## glm(formula = formula_rl, family = binomial(link = "logit"),
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.0321  -0.2677  -0.1383  -0.0573   3.1975
##
## Coefficients:
##                               Estimate Std. Error z value    Pr(>|z|)
## (Intercept)                 -2.243786   9.026444  -0.249       0.804
## Temperature                  0.398337   0.092548   4.304 0.000016768 ***
## Humidity                    -0.327534   0.065160  -5.027 0.000000499 ***
## Hours.Since.Previous.Failure -0.001716   0.001595  -1.076       0.282
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 299.71  on 442  degrees of freedom
## Residual deviance: 121.23  on 439  degrees of freedom
## AIC: 129.23
##
## Number of Fisher Scoring iterations: 7
```

Y calculamos el pseudo R cuadrado:

```
pr2_rl <- 1 -(rl$deviance / rl$null.deviance)
pr2_rl
```

```
## [1] 0.5954985
```

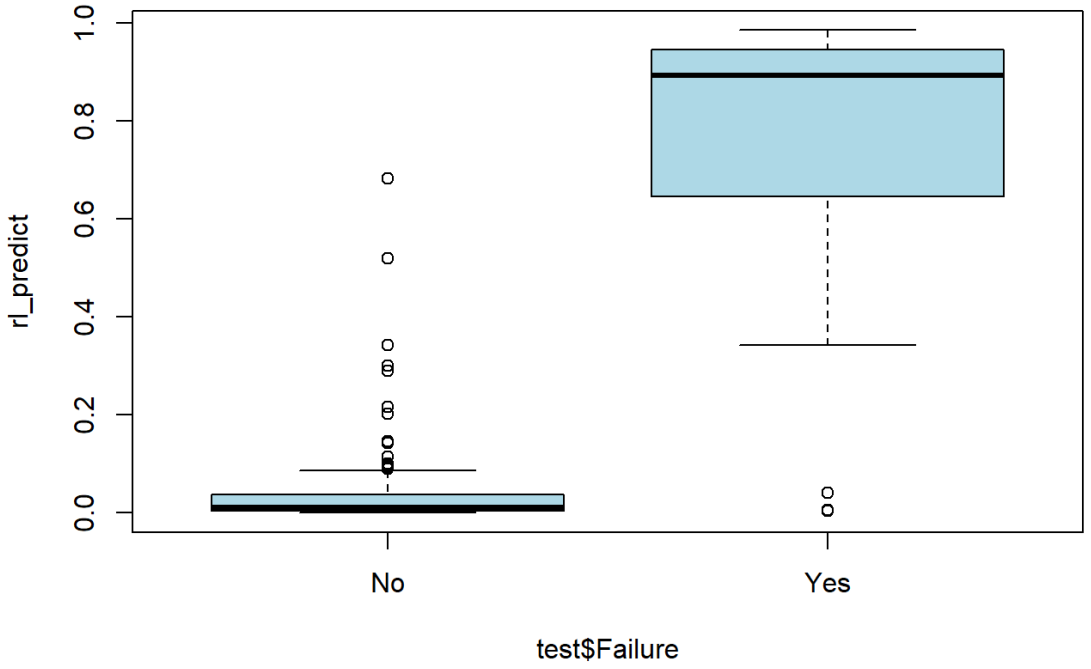
Aplicamos el modelo al conjunto de test, generando un vector con las probabilidades

```
rl_predict<-predict(rl,test,type = 'response')
head(rl_predict)
```

```
##           1           2           3           4           5           6
## 0.010200611 0.040898313 0.002629777 0.051019273 0.082758250 0.008453796
```

Comprobamos en el gráfico:

```
plot(rl_predict~test$Failure, col='lightblue')
```



El modelo está funcionando bien, ya que la probabilidad de que falle la máquina cuando el modelo nos indica que no falla es muy bajo, casi 0, en cambio la probabilidad de que la maqúina falle cuando el modelo indica que falla es muy alto, cerca del 85 %.

Ahora tenemos que transformar la probabilidad en una decisión de si la máquina va a fallar o no.

Con la función umbrales probamos diferentes cortes:

```
umb_rl<-umbrales(test$Failure,rl_predict)
umb_rl
```

```
##    umbral  acierto precision cobertura       F1
## 1    0.05 82.12291  34.09091  83.33333 48.38710
## 2    0.10 92.73743  60.00000  83.33333 69.76744
## 3    0.15 94.41341  68.18182  83.33333 75.00000
## 4    0.20 94.41341  68.18182  83.33333 75.00000
## 5    0.25 95.53073  75.00000  83.33333 78.94737
## 6    0.30 96.08939  78.94737  83.33333 81.08108
## 7    0.35 96.64804  87.50000  77.77778 82.35294
## 8    0.40 96.64804  87.50000  77.77778 82.35294
## 9    0.45 96.64804  87.50000  77.77778 82.35294
## 10   0.50 96.64804  87.50000  77.77778 82.35294
## 11   0.55 97.20670  93.33333  77.77778 84.84848
## 12   0.60 97.20670  93.33333  77.77778 84.84848
## 13   0.65 96.64804  92.85714  72.22222 81.25000
## 14   0.70 97.20670 100.00000  72.22222 83.87097
## 15   0.75 97.20670 100.00000  72.22222 83.87097
## 16   0.80 96.08939 100.00000  61.11111 75.86207
## 17   0.85 96.08939 100.00000  61.11111 75.86207
## 18   0.90 94.41341 100.00000  44.44444 61.53846
## 19   0.95 91.62011 100.00000  16.66667 28.57143
```

Seleccionamos el umbral que maximiza la F1

```
umbral_final_rl<-umb_rl[which.max(umb_rl$F1),1]
umbral_final_rl
```

```
## [1] 0.55
```

Evaluamos la matriz de confusión y las métricas con el umbral optimizado

```
confusion(test$Failure,rl_predict,umbral_final_rl)
```

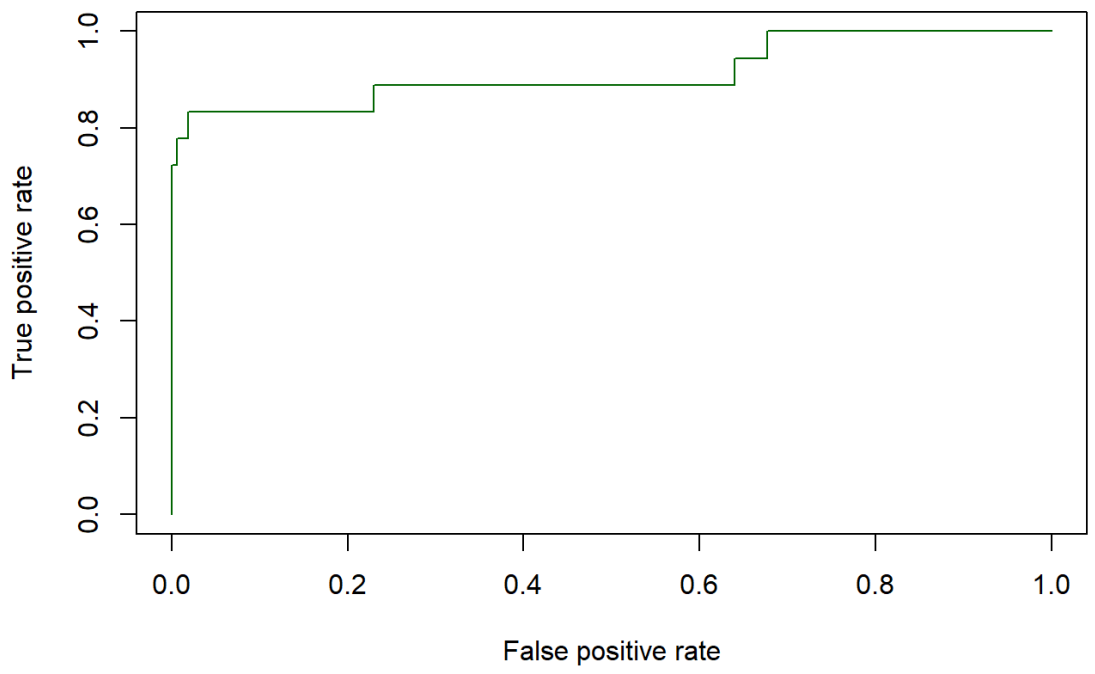
```
##
## real  FALSE TRUE
##   No    160    1
##   Yes     4   14
```

```
rl_metricas<-filter(umb_rl,umbral==umbral_final_rl)
rl_metricas
```

```
##   umbral acierto precision cobertura       F1
## 1   0.55 97.2067  93.33333  77.77778 84.84848
```

Evaluamos la ROC

```
rl_prediction<-prediction(rl_predict,test$Failure)
roc(rl_prediction)
```



Sacamos las métricas definitivas incluyendo el AUC

```
rl_metricas<-cbind(rl_metricas,AUC=round(auc(rl_prediction),2)*100)
print(t(rl_metricas))
```

```
##               [,1]
## umbral     0.55000
## acierto   97.20670
## precision 93.33333
## cobertura 77.77778
## F1        84.84848
## AUC       91.00000
```

Vemos que nos da un valor de AUC por encima del 90 %, lo que nos dice que es un modelo funciona bien.

Aplicamos el modelo al conjutno de datos completo.

```
df$scoring <- predict(rl,df,type='response')
```

Vamos tomar una decisión de si hay que establecer un mantenimiento preventivo por parte de un operario, y elegimos que nos indique cuando va a ocurrir fallo para un scoring superior al 80 %.

Si el scoring es superior al 80 %, pensamos que la máquina va a fallar.

```
df$prediccion <- ifelse(df$scoring > 0.8,1,0)
table(df$prediccion)
```

```
##
##    0    1
## 6985   41
```

Vamos a contrastar la predicción contra la realidad

```
table(df$prediccion,df$Failure)
```

```
##
##       No  Yes
##   0 6961   24
##   1    0   41
```

Vemos que el número de casos en los que el modelo dice que falla pero no se produce este fallo es cero, por lo tanto podemos bajar el umbral y permitir que el modelo aumente sus fallos sin suponer esto un coste elevado para la empresa.

```
df$prediccion <- ifelse(df$scoring > 0.6,1,0)
```

Vamos a contrastar la predicción contra la realidad

```
table(df$prediccion,df$Failure)
```

```
##
##       No  Yes
##   0 6957   16
##   1    4   49
```

Vemos que el número de veces que el modelo dice que falla pero realmente no lo hace se ha incrementado levemente, siendo aún sostenible, e incluso al número de veces que el modelo dice que no falla y finalmente si falla también disminuye, lo que produce una disminución de casos en los que se envíe una persona de mantenimiento y no falle finalmente la máquina.

5 Conclusiones

Se ha trabajado sobre un histórico de mediciones de sensores y se ha obtenido un modelo predictivo de alta calidad.

El modelo es estable y consigue plasmar las características de los momentos en los que se producen los fallos de la máquina.

6 Resultados

Nuestra previsión es que usando este modelo se puede construir un plan de mantenimiento preventivo sobre la maquinaria y predecir cada cuantas horas hay que revisarles o cambiar piezas susceptibles de avería.