Imputació de dades faltants

Arnau Gómez Català

2025-10-04

```
library(readr)
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.4.3
```

```
##
## Adjuntando el paquete: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
data <- read_csv("train.csv", col_types = cols(...1 = col_skip()))
```

```
## New names:
## * `` -> `...1`
```

```
for(i in 1:ncol(data)){
  if(is.character(data[[i]])==TRUE){
    data[[i]] = as.factor(data[[i]])
  }
}

data$HasCrCard <- as.factor(data$HasCrCard)
data$IsActiveMember <- as.factor(data$IsActiveMember)
data$SavingsAccountFlag <- as.factor(data$SavingsAccountFlag)
data$Exited <- as.factor(data$Exited)
```

Podrem veure els valors faltants de la nostra base de dades de forma visual:

```
visdat::vis_dat(data)
```

Veiem només tenim dos tipus de variables (Factor i numèriques) però que tenim una barbaritat de NAs a les nostres variables. Anem a arreglar-ho.

Imputació de valors faltants:

Mirarem primerament si hi han valors extrems que s'hagin de considerar NAs per haver estat incorrectament afegits a la base de dades:

```
summary(data)
```

```
##      Tenure          Gender          EducationLevel        LoanStatus
##  Min.   : 0.000   Female:2233   High School :1658   Active loan :1416
##  1st Qu.: 3.000   Male  :2667   Other       : 217   Default risk: 492
##  Median : 5.000   NA's  :2100   Postgraduate: 736   No loan     :2992
##  Mean   : 5.021                 University  :2289   NA's        :2100
##  3rd Qu.: 7.000                 NA's        :2100
##  Max.   :10.000
##  NA's   :2100
##  NetPromoterScore TransactionFrequency    Surname          Age
##  Min.   : 0.000   Min.   :13.00        Walker :  21   Min.   :18.00
##  1st Qu.: 4.000   1st Qu.:26.00        Scott  :  17   1st Qu.:32.00
##  Median : 8.000   Median :30.00        Martin :  16   Median :37.00
##  Mean   : 6.482   Mean   :30.06        Smith  :  16   Mean   :39.01
##  3rd Qu.: 9.000   3rd Qu.:34.00        Graham :  15   3rd Qu.:44.00
##  Max.   :10.000   Max.   :52.00        (Other):4815   Max.   :92.00
```

```
##  NA's   :2100     NA's   :2100         NA's   :2100   NA's   :2100
##    Geography    ComplaintsCount  HasCrCard   EstimatedSalary     IsActiveMember
##  France :2445   Min.   :0.0000   0   :1462   Min.   :    11.58   0   :2357
##  Germany:1232   1st Qu.:0.0000   1   :3438   1st Qu.: 51271.41   1   :2543
##  Spain  :1223   Median :0.0000   NA's:2100   Median :100218.21   NA's:2100
##  NA's   :2100   Mean   :0.3588               Mean   :100405.20
##                 3rd Qu.:0.0000               3rd Qu.:149613.87
##                 Max.   :5.0000               Max.   :199862.75
##                 NA's   :2100                 NA's   :2100
##  AvgTransactionAmount       CustomerSegment  MaritalStatus
##  Min.   : 19.60       Affluent      :1415   Divorced: 711
##  1st Qu.: 70.19       High Net Worth: 932   Married :2538
##  Median : 98.60       Mass Market   :2553   Single  :1427
##  Mean   :111.35       NA's          :2100   Widowed : 224
##  3rd Qu.:137.32                             NA's    :2100
##  Max.   :581.79
##  NA's   :2100
##  DigitalEngagementScore       ID        CreditScore   SavingsAccountFlag
##  Min.   :  5.00         Min.   :   1   Min.   :350    0   :1650
##  1st Qu.: 50.00         1st Qu.:1760   1st Qu.:583    1   :3250
##  Median : 60.00         Median :3530   Median :651    NA's:2100
##  Mean   : 59.61         Mean   :3515   Mean   :650
##  3rd Qu.: 70.00         3rd Qu.:5266   3rd Qu.:717
##  Max.   :100.00         Max.   :6999   Max.   :850
##  NA's   :2100           NA's   :2100   NA's   :2100
##     Balance       NumOfProducts  Exited
##  Min.   :     0   Min.   :1.00   0:5550
##  1st Qu.:     0   1st Qu.:1.00   1:1450
##  Median : 97576   Median :1.00
##  Mean   : 76733   Mean   :1.53
##  3rd Qu.:127652   3rd Qu.:2.00
##  Max.   :250898   Max.   :4.00
##  NA's   :2100     NA's   :2100
```

Busquem valors tant extrems que siguin irreals i s'hagin de considerar faltants; algú amb 500 anys, una satifacció del client de 13 cuan la variable es mou entre el 0 i el 10, ingresos negatius o variables binàries amb valors diferents a 1 o 0. Per aquesta base de dades no hem trobat valors d'aquest tipus, per tant començarem directament amb la imputació dels valors que ja tenim faltants.

Comprovarem la aleatòrietat de les dades faltants amb el test de Little per a MCAR:

```
naniar::mcar_test(data)
```

```
## # A tibble: 1 x 4
##   statistic     df p.value missing.patterns
##       <dbl>  <dbl>   <dbl>            <int>
## 1   112066. 112035   0.473             6858
```

Veiem que per un P-valor de 0.4730255 els nostres Missings are Completly generated At Random. Llavors no ens haurem de preocupar de trobar patrons per els cuals hagin sigut generats.

Comencem amb la imputació principalment per tres mètodes; **MICE**, **Veïns Propers** i **Bosc Aleatòri**. D'aquesta forma tindrem 3 possibles bases de dades que podem utilitzar per al model predictiu i podrem escollir quin dels tres mètodes funcionarà millor, fent que tinguem un *F1 score* major.

MICE:

Podrem fer la primera imputació de les dades faltants a través de l'algorisme MICE (Multiple Imputation by Chained Equations).

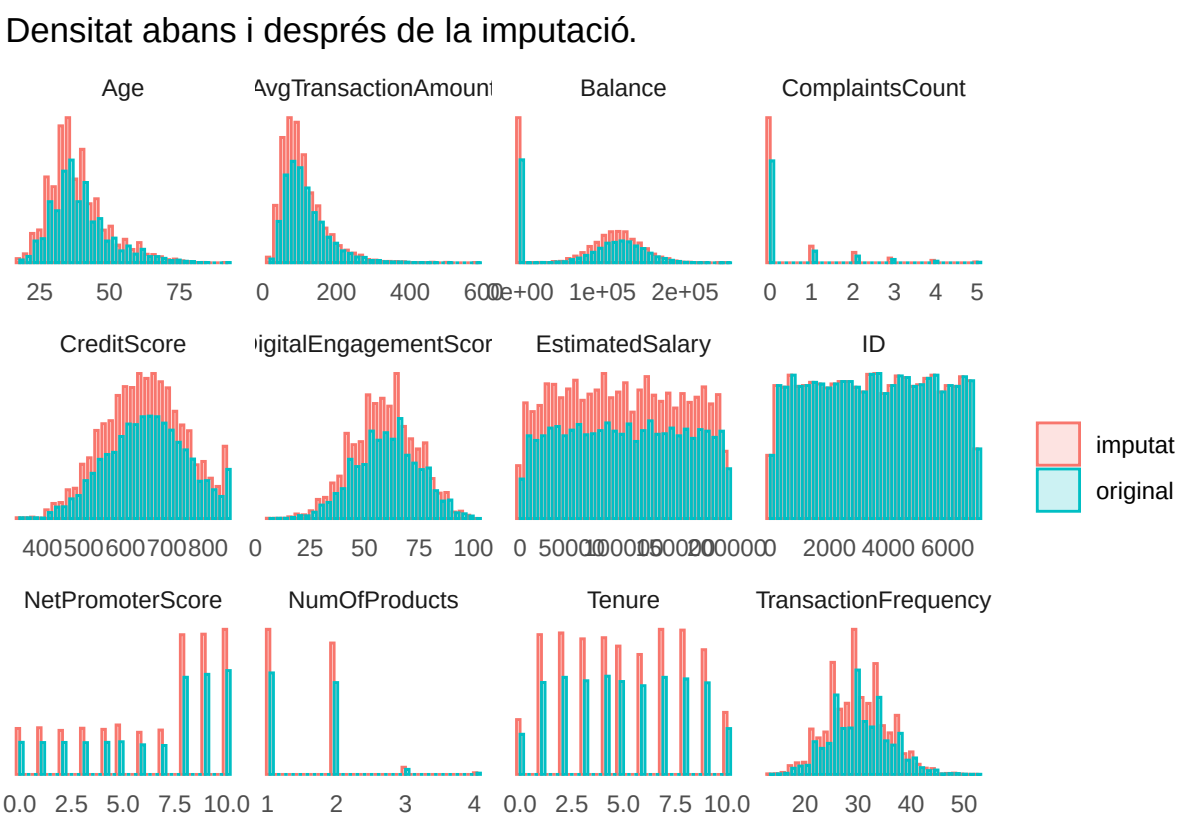
- Utilitzarem el mètode *pmm* (Predictive Mean Matching), per les variables numèriques, excepte *ID*.
- El mètode *logreg* (Regressió Logística), per les variables binàries, excepte *Exited*.
- Y per últim el mètode *polyreg* (Regressió Logística Politòmica), per les variables categòriques amb més d'un factor.

```
density_before_after <- function(before, after) {
    require(ggplot2)

    density_df_before <- before |>
        select(where(is.numeric)) |>
        mutate(imputation = "original")

    density_df_after <- after |>
        select(where(is.numeric)) |>
        mutate(imputation = "imputat")

    density_df <- bind_rows(density_df_before, density_df_after) |>
        mutate(imputation = factor(imputation)) |>
        tidyr::pivot_longer(!imputation, names_to = "variable") |>
        filter(!is.na(value))

    ggplot(density_df, aes(x = value, color = imputation, fill = imputation)) +
        facet_wrap(~variable, scales = "free") +
        geom_histogram(alpha = 0.2, width = 0.1, bins = 30, position = "dodge") +
        scale_color_discrete(aesthetics = c("color", "fill"), name = "") +
        scale_y_continuous(breaks = NULL) +
        xlab("") +
        ylab("") +
        ggtitle("Densitat abans i després de la imputació.") +
        theme_minimal() +
        theme(panel.grid = element_blank())
}
mass_before_after <- function(before, after) {
    require(ggplot2)

    mass_df_before <- before |>
        select(where(is.factor)) |>
        mutate(imputation = "original")

    mass_df_after <- after |>
        select(where(is.factor)) |>
        mutate(imputation = "imputat")

    levels_in_order <- lapply(mass_df_before, levels) |> unlist()

    mass_df <- bind_rows(mass_df_before, mass_df_after) |>
      mutate(imputation = factor(imputation)) |>
      tidyr::pivot_longer(!imputation, names_to = "variable") |>
      filter(!is.na(value)) |>
```

```
      group_by(imputation, variable) |>
      reframe(prop = proportions(table(value)), category = names(prop)) |>
      group_by(variable) |>
      mutate(category = factor(category, levels = levels(mass_df_before[[unique(variable)]]))) |>
      filter(prop > 0)

    ggplot(mass_df, aes(x = category, y = prop, color = imputation, fill = imputation)) +
        facet_wrap(~variable, scales = "free") +
        geom_col(alpha = 0.2, width = 0.6, position = "dodge") +
        scale_color_discrete(aesthetics = c("color", "fill"), name = "") +
        scale_y_continuous(breaks = NULL) +
        xlab("") +
        ylab("") +
        ggtitle("Massa abans i després de la imputació") +
        theme_minimal() +
        theme(panel.grid = element_blank(),
              axis.text.x = element_text(angle = 45, hjust = 1))
}
```

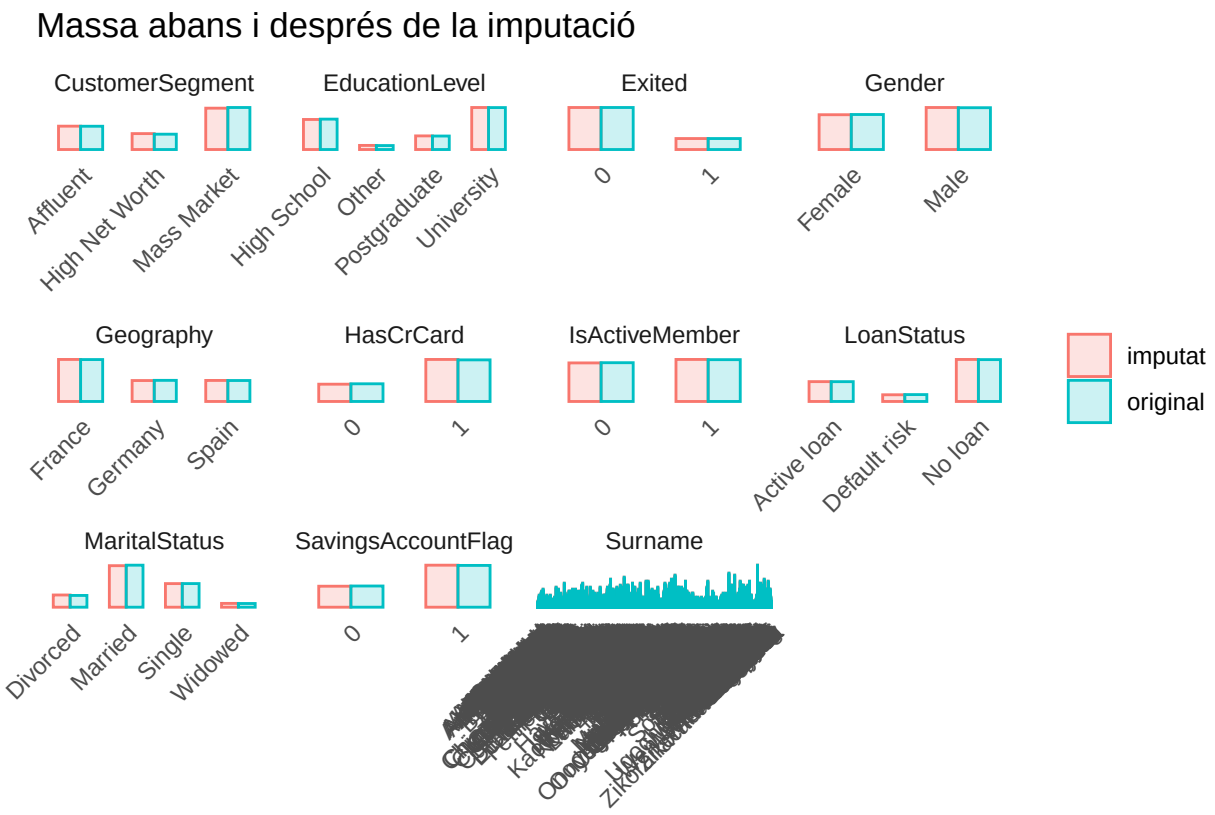
```
density_before_after(data, data_imputed_MICE)
```

```
## Cargando paquete requerido: ggplot2
```



Sembla que les deistribucions de les dades faltants és clavada a la de les dades originals, cosa molt positiva però les freqüències són majors per a les dades faltants. Aquest fet no serà un problema per al nostre objectiu predictiu.

```
mass_before_after(data, data_imputed_MICE)
```



Per a les variables factor, tant binàries com politòmiques, tenim que s'ha imputat correctíssimament tant les deistribucions com les freqüències. L'única variable a menyprear serà *Surname* que al ser-hi una variable amb molts nivells, no ho fa correctament i obviarem en el nostre model predictiu, no ens sembla que sigui rellevant per a predir si marxarà o no una persona del nostre banc.

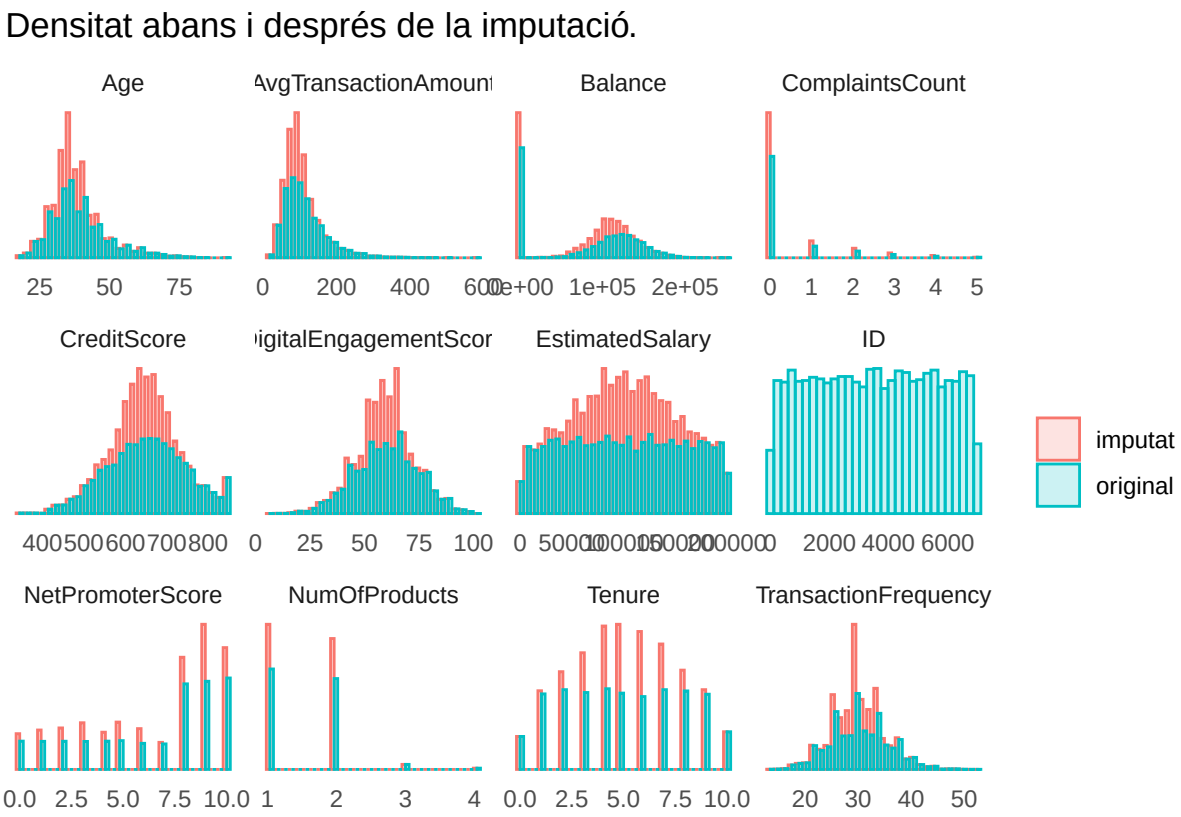
Veïns Propers:

Imputarem els valors faltants dels 7 veïns (Número) més propers, utilitzarem la mitjana d'aquests veïns per imputar les variables numèriques i utilitzarem la moda d'aquest veïns per les variables categòriques.

```
data_for_imputation <- cbind(data_for_imputation_bin, data_for_imputation_fac, data_for_imputation_num)


set.seed(28657)
imputed_data <- VIM::kNN(
    as.data.frame(data_for_imputation),
    k = 7, imp_var = FALSE)

data_imputed_KNN <- complete_columns(data_for_imputation, imputed_data)
```

Podrem comprarar-ho de la mateixa forma que amb l'algorisme anterior:

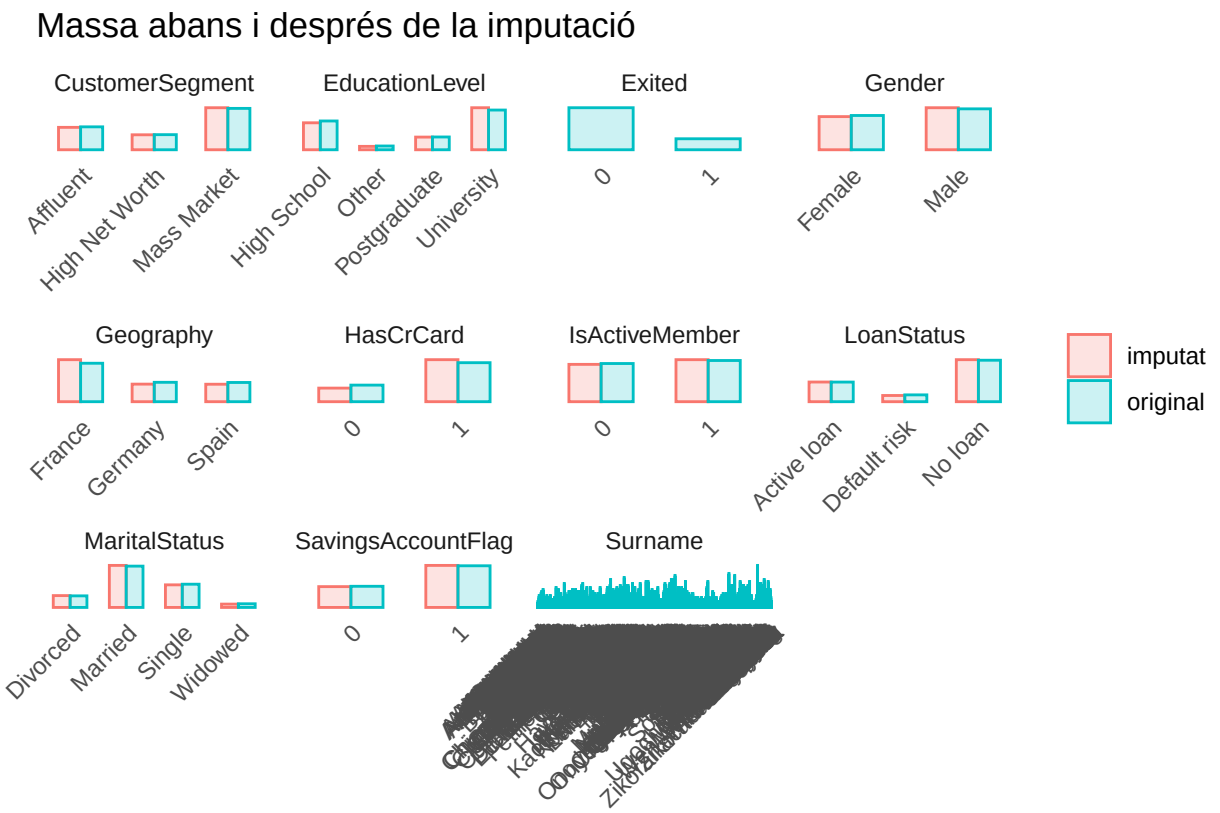
```
density_before_after(data, data_imputed_KNN)
```



Veiem que ara les imputacions de dades faltants no segueixen les distribucions de les dades orignials, això significa que no han sigut correctament imputades i podriem descartar aquest mètode d'imputació, almenys per les variables numériques.

Veiem com ho fa per les varibles categòriques:

```
mass_before_after(data, data_imputed_KNN)
```



Per aquestes veiem que si han sigut millors les imputacions, tot i que sempre hi ha un nivell que queda sobrerepresentat en comparació amb la resta distorcionant lleugerament les distribucions. Ho haurem de tenir en compte a l'hora d'escollir model.

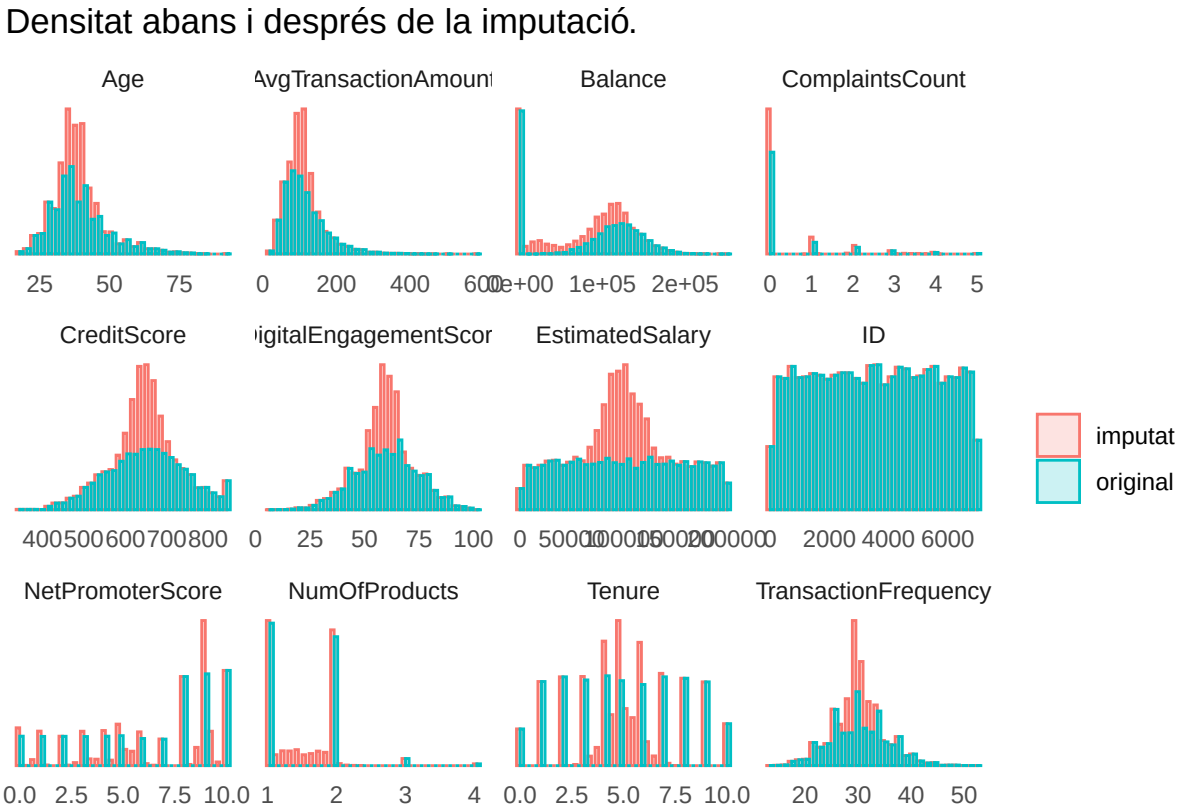
Bosc Aleatòri:

```
bind_cols(
    variable = names(forest_output$ximp),
    tipus_error = names(forest_output$OOBerror),
    error = round(forest_output$OOBerror, 2)
) |>
    arrange(tipus_error, -error) |>
    knitr::kable(col.names = c("Variable", "Tipus d'Error", "Error"))
```

Variable	Tipus d'Error	Error
EstimatedSalary	MSE	3.591450e+09
Balance	MSE	2.225580e+09
CreditScore	MSE	1.024053e+04
AvgTransactionAmount	MSE	3.839890e+03
DigitalEngagementScore	MSE	2.143000e+02
Age	MSE	1.226300e+02
TransactionFrequency	MSE	3.384000e+01
Tenure	MSE	9.110000e+00

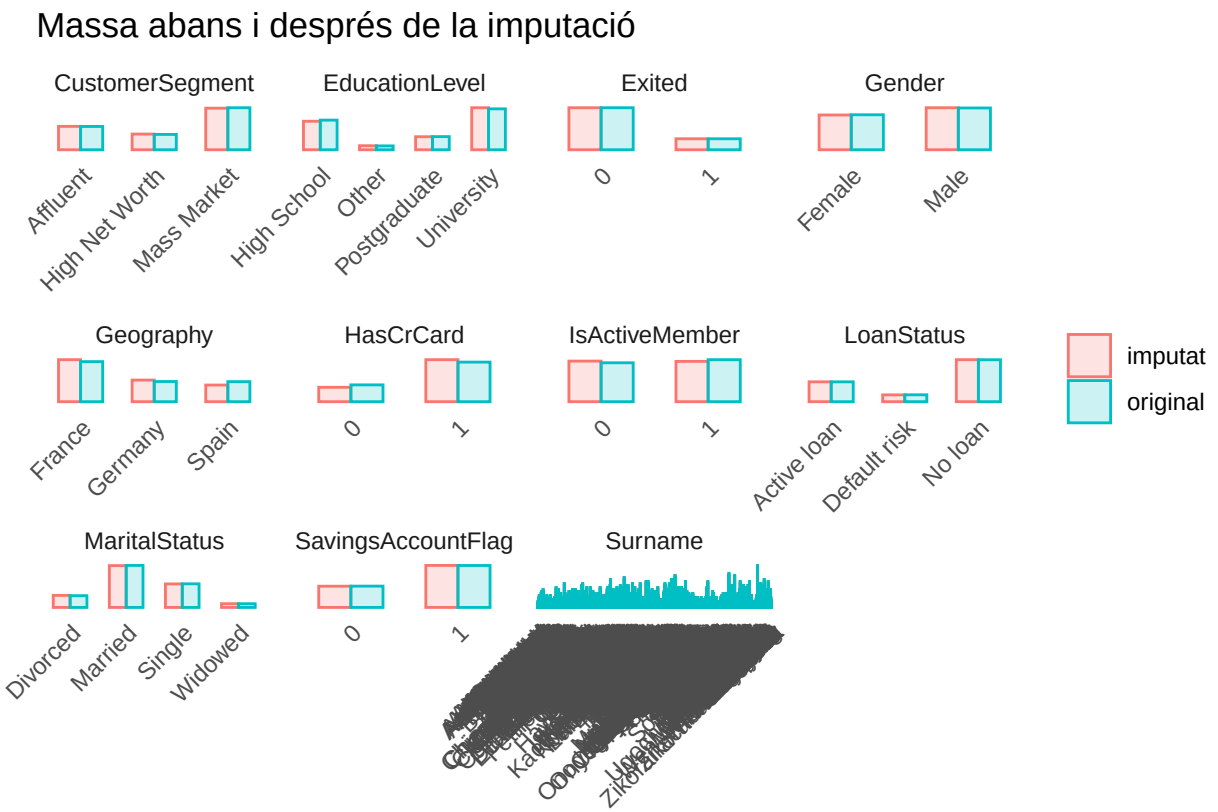
Variable	Tipus d'Error	Error
NetPromoterScore	MSE	2.080000e+00
NumOfProducts	MSE	3.000000e-01
ComplaintsCount	MSE	3.000000e-02
Gender	PFC	4.800000e-01
IsActiveMember	PFC	4.600000e-01
Geography	PFC	4.600000e-01
HasCrCard	PFC	3.900000e-01
EducationLevel	PFC	6.000000e-02
LoanStatus	PFC	3.000000e-02
SavingsAccountFlag	PFC	2.000000e-02
CustomerSegment	PFC	0.000000e+00
MaritalStatus	PFC	0.000000e+00

```
density_before_after(data, data_imputed_FOREST)
```



Veiem que aquest mètode és el pitjor dels utilitzats prèviament, sembla ajuntar tots els valors imputats al centre, al voltant de la mitjana de cada variable en una mena de distribució normal forzada que no ens ajudarà pas en el nostre objectiu.

```
mass_before_after(data, data_imputed_FOREST)
```



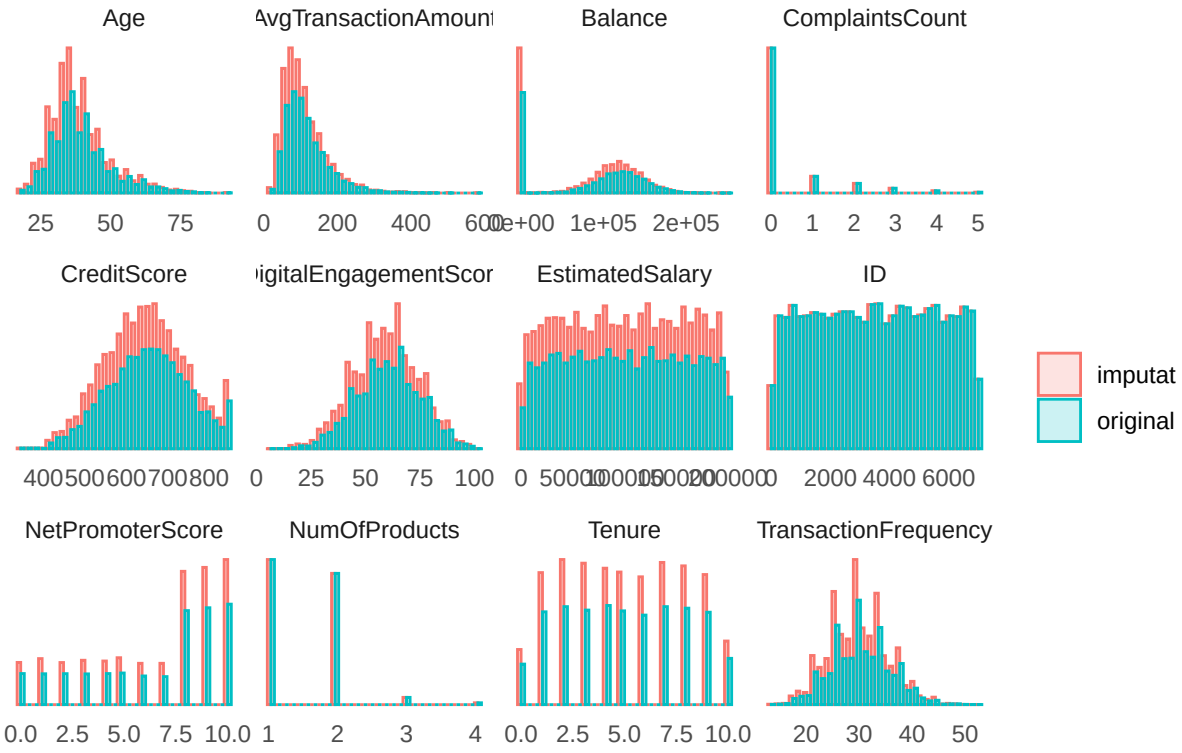
Encara que es cert que en les variables categòriques aconsegueix una imputació molt semblant al *MICE* i millor que amb els *K-Nearest Neighbors*.

Imputació per Bootstrap:

Amb la funció *aregImpute()* podrem fer imputació de les dades faltants basada en regressions amb splines. Utilitza regressió aditiva amb splines per a variables numèriques i regressió logística per a les categòriques.

```
density_before_after(data, data_imputed_AREG)
```

Densitat abans i després de la imputació.




```
mass_before_after(data, data_imputed_AREG)
```

Massa abans i després de la imputació