Rental-Price-Classification

Paula Ramirez - Student id 8963215

01/12/2024

Rental-Price-Classification - PART A

1. Preliminary Data Preparation

1. Rename all variables

Appending my initials to all variables

```
#Reading the file
data_lease_PR <- read.table(here("Rental-Price-Classification", "Rent_Class.txt"),
                                 header = TRUE, sep = ",")
#Converting it to dataframe.
data_lease_PR <- as.data.frame(data_lease_PR)
#Append PR initials to all variables in the dataframe
colnames(data_lease_PR) <- paste(colnames(data_lease_PR), "PR", sep = "_")
#Changing to factor
data_lease_PR <-as.data.frame(unclass(data_lease_PR), stringsAsFactors = TRUE)
str(data_lease_PR)
```

```
## 'data.frame':    1051 obs. of  9 variables:
##  $ Prc_PR     : int  1590 2460 2630 2820 2740 2370 1870 1850 1750 2090 ...
##  $ Bed_PR     : int  3 2 1 1 4 3 1 3 4 1 ...
##  $ floor_PR   : int  3 5 4 2 1 5 1 2 1 1 ...
##  $ TotFloor_PR: int  4 7 6 4 2 5 4 4 5 4 ...
##  $ Bath_PR    : int  4 2 2 1 3 2 1 1 1 1 ...
##  $ Sqft_PR    : int  501 1275 982 2418 1655 1024 864 671 609 834 ...
##  $ City_PR    : Factor w/ 3 levels "Blossomville",..: 1 3 2 3 3 2 3 1 2 3 ...
##  $ Comp_PR    : Factor w/ 2 levels "Leaseflow","Rentopia": 1 2 1 1 2 1 2 2 2 2 ...
##  $ Dist_PR    : num  10.6 1.4 6.8 0.4 5.3 0.5 3.5 3.6 1.4 4 ...
```

```
#Showing first results
head(data_lease_PR)
```

```
##   Prc_PR Bed_PR floor_PR TotFloor_PR Bath_PR Sqft_PR      City_PR   Comp_PR
## 1   1590      3        3           4       4     501 Blossomville Leaseflow
## 2   2460      2        5           7       2    1275    Terranova  Rentopia
## 3   2630      1        4           6       2     982    Riverport Leaseflow
## 4   2820      1        2           4       1    2418    Terranova Leaseflow
## 5   2740      4        1           2       3    1655    Terranova  Rentopia
```

```
## 6   2370      3        5           5       2    1024    Riverport Leaseflow
##   Dist_PR
## 1    10.6
## 2     1.4
## 3     6.8
## 4     0.4
## 5     5.3
## 6     0.5
```

2. Graphical Data Summaries

Showing simple summaries, there was no need showing plots due to there are not outliers.

```
# Showing simple summaries
round(stat.desc(data_lease_PR),2)
```

```
##                  Prc_PR  Bed_PR floor_PR TotFloor_PR Bath_PR    Sqft_PR City_PR
## nbr.val         1051.00 1051.00  1051.00     1051.00 1051.00    1051.00      NA
## nbr.null           0.00    0.00     0.00        0.00    0.00       0.00      NA
## nbr.na             0.00    0.00     0.00        0.00    0.00       0.00      NA
## min              507.00    1.00     1.00        1.00    1.00     501.00      NA
## max             7480.00    4.00    10.00       10.00    4.00    3254.00      NA
## range           6973.00    3.00     9.00        9.00    3.00    2753.00      NA
## sum          2365741.00 2236.00  2388.00     4097.00 1701.00 1355472.00      NA
## median          2110.00    2.00     2.00        4.00    1.00    1231.00      NA
## mean            2250.94    2.13     2.27        3.90    1.62    1289.70      NA
## SE.mean           32.37    0.04     0.05        0.06    0.03      16.98      NA
## CI.mean.0.95      63.51    0.07     0.10        0.13    0.06      33.32      NA
## var          1101101.31    1.37     2.77        4.33    0.96  303065.86      NA
## std.dev         1049.33    1.17     1.66        2.08    0.98     550.51      NA
## coef.var           0.47    0.55     0.73        0.53    0.60       0.43      NA
##              Comp_PR Dist_PR
## nbr.val           NA 1051.00
## nbr.null          NA    1.00
## nbr.na            NA    0.00
## min               NA    0.00
## max               NA   21.00
## range             NA   21.00
## sum               NA 4319.60
## median            NA    3.50
## mean              NA    4.11
## SE.mean           NA    0.09
## CI.mean.0.95      NA    0.18
## var               NA    8.60
## std.dev           NA    2.93
## coef.var          NA    0.71
```

3. Training and Test Set

The rate of data for my train and test set is 75/25 My speed is –> 3215 (student id)

```r
# Number of rows of data
n.row <- nrow(data_lease_PR)
# Choosing speed
set.seed(3215)
# Choosing Sampling rate
sr_pr <- 0.75
#Choose the rows for the training sample with my student id
training.rows <- sample(1:n.row, sr_pr*n.row, replace=FALSE)
#Assign to the training sample
train_pr <- subset(data_lease_PR[training.rows,])
# Assign the balance to the Test Sample (rest of data)
test_pr <- subset(data_lease_PR[-c(training.rows),])
```

The next code will compare the train and test dataset

```r
#summaries
summary(train_pr)
```

```
##      Prc_PR         Bed_PR         floor_PR       TotFloor_PR
##  Min.   : 507   Min.   :1.000   Min.   : 1.000   Min.   : 1.00
##  1st Qu.:1600   1st Qu.:1.000   1st Qu.: 1.000   1st Qu.: 2.00
##  Median :2120   Median :2.000   Median : 2.000   Median : 4.00
##  Mean   :2259   Mean   :2.128   Mean   : 2.259   Mean   : 3.89
##  3rd Qu.:2690   3rd Qu.:3.000   3rd Qu.: 3.000   3rd Qu.: 5.00
##  Max.   :6840   Max.   :4.000   Max.   :10.000   Max.   :10.00
##     Bath_PR         Sqft_PR              City_PR         Comp_PR
##  Min.   :1.000   Min.   : 505.0   Blossomville:278   Leaseflow:259
##  1st Qu.:1.000   1st Qu.: 776.5   Riverport   :264   Rentopia :529
##  Median :1.000   Median :1223.0   Terranova   :246
##  Mean   :1.636   Mean   :1288.4
##  3rd Qu.:2.000   3rd Qu.:1656.0
##  Max.   :4.000   Max.   :3254.0
##     Dist_PR
##  Min.   : 0.00
##  1st Qu.: 2.00
##  Median : 3.50
##  Mean   : 4.17
##  3rd Qu.: 5.50
##  Max.   :21.00
```

```r
summary(test_pr)
```

```
##      Prc_PR         Bed_PR         floor_PR      TotFloor_PR
##  Min.   : 509   Min.   :1.000   Min.   :1.000   Min.   : 1.000
##  1st Qu.:1510   1st Qu.:1.000   1st Qu.:1.000   1st Qu.: 2.000
##  Median :2090   Median :2.000   Median :2.000   Median : 4.000
##  Mean   :2227   Mean   :2.125   Mean   :2.312   Mean   : 3.924
##  3rd Qu.:2590   3rd Qu.:3.000   3rd Qu.:3.000   3rd Qu.: 5.000
##  Max.   :7480   Max.   :4.000   Max.   :9.000   Max.   :10.000
##     Bath_PR         Sqft_PR             City_PR        Comp_PR
##  Min.   :1.000   Min.   : 501   Blossomville:87   Leaseflow:103
##  1st Qu.:1.000   1st Qu.: 829   Riverport   :91   Rentopia :160
```

```
##  Median :1.000   Median :1252   Terranova   :85
##  Mean   :1.567   Mean   :1293
##  3rd Qu.:2.000   3rd Qu.:1618
##  Max.   :4.000   Max.   :2804
##     Dist_PR
##  Min.   : 0.100
##  1st Qu.: 1.800
##  Median : 3.500
##  Mean   : 3.929
##  3rd Qu.: 5.300
##  Max.   :17.000
```

```
# mean in Prc_PR each set
round(mean(train_pr$Prc_PR),6)
```

```
## [1] 2258.816
```

```
round(mean(test_pr$Prc_PR),6)
```

```
## [1] 2227.354
```

```
#commparing means with t test
t.test(train_pr$Prc_PR, test_pr$Prc_PR)
```

```
##
##  Welch Two Sample t-test
##
## data:  train_pr$Prc_PR and test_pr$Prc_PR
## t = 0.40976, df = 429.82, p-value = 0.6822
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -119.4545  182.3793
## sample estimates:
## mean of x mean of y
##  2258.816  2227.354
```

In the summaries, I did not evidence any dissimilarities. The means in **Prc_PR** show that there are not significant differences between sets.

In addition, the test shows that p-value is equal to 0.68 (> 0.05), indicating that we can't reject the null hypothesis that means are equal. Based on these findings, it is appropriate to proceed with these datasets as they are comparable.

4. Creating a new Variable in test and train

```
# Creating new variable based on Prc
train_pr$PC_PR <- as.factor(ifelse(train_pr$Prc_PR < 2251,"L","H"))
test_pr$PC_PR <- as.factor(ifelse(test_pr$Prc_PR < 2251,"L","H"))
# Deleting Prc variable
train_pr <- train_pr[, -c(1)]
test_pr <- test_pr[, -c(1)]
head(train_pr)
```

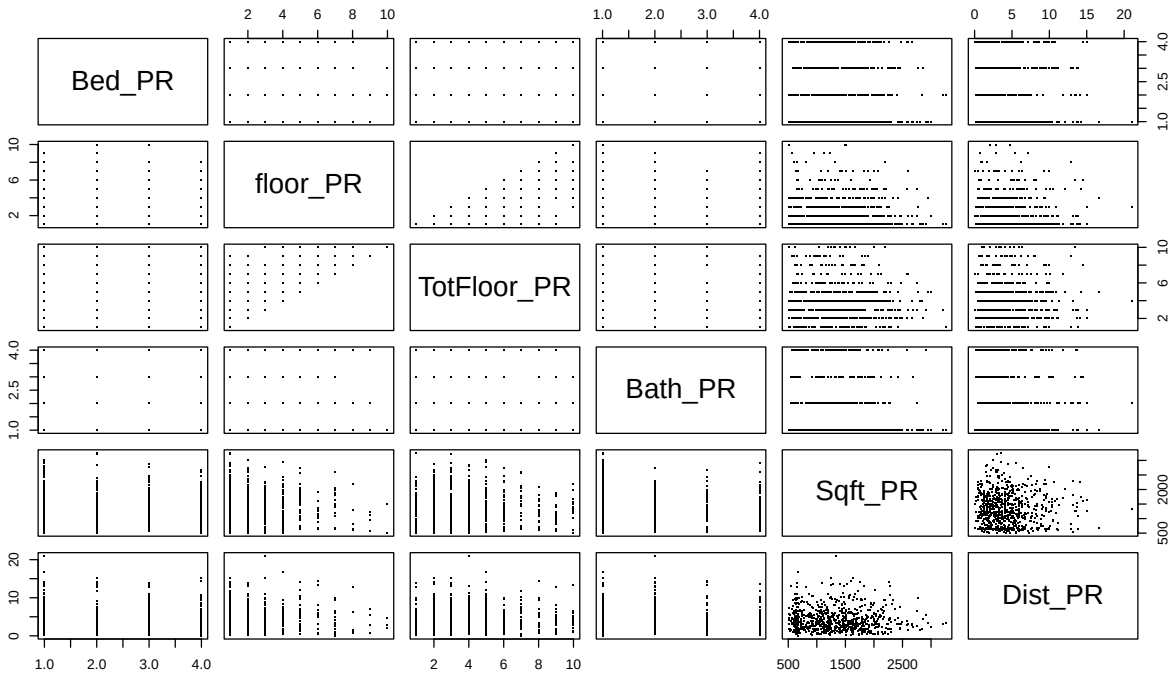
```
##     Bed_PR floor_PR TotFloor_PR Bath_PR Sqft_PR      City_PR   Comp_PR Dist_PR
## 21       3        1           3       1    1730    Riverport  Rentopia     1.6
## 387      1        1           2       1     919 Blossomville  Rentopia     1.1
## 924      1        3           4       1    1124    Terranova  Rentopia     9.0
## 640      2        1           1       1     913    Terranova Leaseflow     5.3
## 64       1        1           2       1    2668 Blossomville  Rentopia     2.9
## 737      3        1           1       2    1076    Riverport  Rentopia     9.3
##     PC_PR
## 21      H
## 387     H
## 924     L
## 640     L
## 64      H
## 737     L
```

The new categorical variable called PC_PR was created in both train and test dataset. Income values higher than 2251 are marked as "H" and "L" otherwise. I have deleted the Prc_PR as requested.

2. Exploratory Analysis

1. Correlations

```
# correlations only with numerical variables
pairs(train_pr[sapply(train_pr,is.numeric)], pch=46)
```



```r
# correlations with factors (polycor library)
corr_data_pr <- hetcor(train_pr)
round(corr_data_pr$correlations,2)
```

```
##             Bed_PR floor_PR TotFloor_PR Bath_PR Sqft_PR City_PR Comp_PR Dist_PR
## Bed_PR        1.00     0.03       -0.02   -0.09   -0.02   -0.07    0.03   -0.05
## floor_PR      0.03     1.00        0.68    0.00   -0.02    0.07   -0.04    0.01
## TotFloor_PR  -0.02     0.68        1.00   -0.01    0.00    0.03    0.03    0.00
## Bath_PR      -0.09     0.00       -0.01    1.00   -0.04    0.01    0.02    0.08
## Sqft_PR      -0.02    -0.02        0.00   -0.04    1.00   -0.05    0.01    0.01
## City_PR      -0.07     0.07        0.03    0.01   -0.05    1.00   -0.01    0.04
## Comp_PR       0.03    -0.04        0.03    0.02    0.01   -0.01    1.00   -0.01
## Dist_PR      -0.05     0.01        0.00    0.08    0.01    0.04   -0.01    1.00
## PC_PR         0.41    -0.05       -0.02   -0.37   -0.23    0.10    0.00    0.24
##             PC_PR
## Bed_PR       0.41
## floor_PR    -0.05
## TotFloor_PR -0.02
## Bath_PR     -0.37
## Sqft_PR     -0.23
## City_PR      0.10
## Comp_PR      0.00
## Dist_PR      0.24
## PC_PR        1.00
```

```r
#print correlations types
print(corr_data_pr)
```

```
##
## Two-Step Estimates
##
## Correlations/Type of Correlation:
##               Bed_PR  floor_PR TotFloor_PR  Bath_PR  Sqft_PR    City_PR
## Bed_PR             1   Pearson     Pearson  Pearson  Pearson Polyserial
## floor_PR     0.02986         1     Pearson  Pearson  Pearson Polyserial
## TotFloor_PR -0.01679    0.6779           1  Pearson  Pearson Polyserial
## Bath_PR     -0.09154 -0.001312   -0.007924        1  Pearson Polyserial
## Sqft_PR     -0.01529  -0.02265    0.002553 -0.04499        1 Polyserial
## City_PR     -0.07335   0.06525      0.0319 0.009451 -0.04817          1
## Comp_PR      0.03346  -0.04168     0.02994  0.01657   0.0106  -0.007077
## Dist_PR      -0.0479   0.01484    0.000816   0.0835  0.01108    0.03919
## PC_PR         0.4121  -0.05168    -0.02312  -0.3712  -0.2263    0.09725
##                Comp_PR    Dist_PR      PC_PR
## Bed_PR      Polyserial    Pearson Polyserial
## floor_PR    Polyserial    Pearson Polyserial
## TotFloor_PR Polyserial    Pearson Polyserial
## Bath_PR     Polyserial    Pearson Polyserial
## Sqft_PR     Polyserial    Pearson Polyserial
## City_PR     Polychoric Polyserial Polychoric
## Comp_PR              1 Polyserial Polychoric
## Dist_PR       -0.01087          1 Polyserial
## PC_PR       -0.0007136     0.2437          1
```

```
##
## Standard Errors:
##              Bed_PR floor_PR TotFloor_PR Bath_PR Sqft_PR City_PR Comp_PR
## Bed_PR
## floor_PR    0.03561
## TotFloor_PR 0.03563  0.01928
## Bath_PR     0.03535  0.03564     0.03564
## Sqft_PR     0.03564  0.03563     0.03564 0.03557
## City_PR     0.03963   0.0399     0.03995 0.04007  0.0399
## Comp_PR     0.04633  0.04574     0.04643 0.04645 0.04655 0.05194
## Dist_PR     0.03556  0.03564     0.03564  0.0354 0.03564 0.03998 0.04631
## PC_PR       0.03659  0.04451     0.04487 0.03809 0.04232  0.0501 0.05831
##             Dist_PR
## Bed_PR
## floor_PR
## TotFloor_PR
## Bath_PR
## Sqft_PR
## City_PR
## Comp_PR
## Dist_PR
## PC_PR       0.04421
##
## n = 788
##
## P-values for Tests of Bivariate Normality:
##                 Bed_PR   floor_PR TotFloor_PR    Bath_PR     Sqft_PR
## Bed_PR
## floor_PR    1.532e-146
## TotFloor_PR 2.951e-130  2.863e-70
## Bath_PR     2.002e-275 9.253e-194   1.71e-176
## Sqft_PR     3.613e-114  2.187e-33   2.099e-18  6.02e-160
## City_PR     9.765e-116  4.379e-34   2.356e-16 1.192e-161    0.003255
## Comp_PR     2.993e-119  1.233e-35   2.469e-18 1.523e-164    0.004579
## Dist_PR     8.382e-118  2.747e-37   1.842e-21 1.005e-163 0.000000652
## PC_PR       2.474e-119   1.18e-36   3.636e-18 8.213e-167    0.002528
##                   City_PR      Comp_PR      Dist_PR
## Bed_PR
## floor_PR
## TotFloor_PR
## Bath_PR
## Sqft_PR
## City_PR
## Comp_PR            0.4385
## Dist_PR     0.00000002538 0.0000004666
## PC_PR         0.000007476         <NA> 0.0000005711
```

Findings The more significant correlations are:

- **TotFloor_PR and floor_PR** 68% of correlation. There is an obvious positive and strong correlation between both variables. Which indicates that buildings with more floors have apartments located in higher floors.
- **Bed_PR and PC_PM** 41% of correlation. Indicates that apartments with more bedrooms tend to have a "High" monthly rent

- **Bath_PR and PC_PM** -37% correlation. There is a moderate and negative correlation, which means that apartments with less bathrooms are *surprisingly* more expensive than apartments with more bathrooms.
- **PC_PM and Sqft_PR** -23% correlation. Same as above, there is an unusual negative correlation between price and apartment size. Meaning that smaller apartments tend to have a higher rent price, which is rare. However the correlation is weak.
- **PC_PM and Dist_PR** 24% correlation. This weak relationship indicates that if there is more distance is between apartments and center of town more expensive apartment is.

The p-values associated with the correlations indicate that the relationships are significant. There is an exception in **City_PR** and **Comp_PR** which has a 0.4385 (a pvalue > 0.05) indicating that correlation are less reliable.

3. Model Development

1. Logistic regression models - Full model

```
# Full Model All variables
rent_glm_full = glm(PC_PR ~ . ,
              family="binomial", data=train_pr, na.action=na.omit)
# Summary
summary(rent_glm_full)
```

```
##
## Call:
## glm(formula = PC_PR ~ ., family = "binomial", data = train_pr,
##     na.action = na.omit)
##
## Coefficients:
##                    Estimate Std. Error z value         Pr(>|z|)
## (Intercept)       0.5670602  0.4101851   1.382         0.166834
## Bed_PR            0.8307329  0.0852788   9.741          < 2e-16 ***
## floor_PR         -0.1319434  0.0723935  -1.823         0.068366 .
## TotFloor_PR       0.0428245  0.0582825   0.735         0.462476
## Bath_PR          -0.8995172  0.1021015  -8.810          < 2e-16 ***
## Sqft_PR          -0.0010257  0.0001640  -6.253 0.00000000040153 ***
## City_PRRiverport -0.8009021  0.2137752  -3.746         0.000179 ***
## City_PRTerranova  0.6611484  0.2215523   2.984         0.002844 **
## Comp_PRRentopia   0.0004503  0.1879458   0.002         0.998088
## Dist_PR           0.2490350  0.0358818   6.940 0.00000000000391 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1078.10  on 787  degrees of freedom
## Residual deviance:  777.69  on 778  degrees of freedom
## AIC: 797.69
##
## Number of Fisher Scoring iterations: 5
```

2. Logistic regression models - Backward model

```
# Backward Model All variables
rent_glm_back <- step(rent_glm_full, direction ="backward", trace=TRUE)
```

```
## Start:  AIC=797.69
## PC_PR ~ Bed_PR + floor_PR + TotFloor_PR + Bath_PR + Sqft_PR +
##     City_PR + Comp_PR + Dist_PR
##
##               Df Deviance    AIC
## - Comp_PR      1   777.69 795.69
## - TotFloor_PR  1   778.23 796.23
## <none>             777.69 797.69
## - floor_PR     1   781.04 799.04
## - City_PR      2   821.28 837.28
## - Sqft_PR      1   819.99 837.99
## - Dist_PR      1   836.17 854.17
## - Bath_PR      1   874.06 892.06
## - Bed_PR       1   893.32 911.32
##
## Step:  AIC=795.69
## PC_PR ~ Bed_PR + floor_PR + TotFloor_PR + Bath_PR + Sqft_PR +
##     City_PR + Dist_PR
##
##               Df Deviance    AIC
## - TotFloor_PR  1   778.24 794.24
## <none>             777.69 795.69
## - floor_PR     1   781.05 797.05
## - City_PR      2   821.33 835.33
## - Sqft_PR      1   820.00 836.00
## - Dist_PR      1   836.19 852.19
## - Bath_PR      1   874.08 890.08
## - Bed_PR       1   893.41 909.41
##
## Step:  AIC=794.24
## PC_PR ~ Bed_PR + floor_PR + Bath_PR + Sqft_PR + City_PR + Dist_PR
##
##            Df Deviance    AIC
## <none>          778.24 794.24
## - floor_PR  1   781.49 795.49
## - City_PR   2   822.24 834.24
## - Sqft_PR   1   820.48 834.48
## - Dist_PR   1   836.67 850.67
## - Bath_PR   1   874.73 888.73
## - Bed_PR    1   893.45 907.45
```

```
# Summary
summary(rent_glm_back)
```

```
##
## Call:
## glm(formula = PC_PR ~ Bed_PR + floor_PR + Bath_PR + Sqft_PR +
```

```
##     City_PR + Dist_PR, family = "binomial", data = train_pr,
##     na.action = na.omit)
##
## Coefficients:
##                    Estimate Std. Error z value         Pr(>|z|)
## (Intercept)       0.6640824  0.3707437   1.791         0.073258 .
## Bed_PR            0.8272978  0.0850260   9.730          < 2e-16 ***
## floor_PR         -0.0960119  0.0530092  -1.811         0.070105 .
## Bath_PR          -0.8984981  0.1019090  -8.817          < 2e-16 ***
## Sqft_PR          -0.0010245  0.0001639  -6.251 0.00000000040717 ***
## City_PRRiverport -0.8119386  0.2131527  -3.809         0.000139 ***
## City_PRTerranova  0.6543915  0.2210927   2.960         0.003078 **
## Dist_PR           0.2492133  0.0359142   6.939 0.00000000000395 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1078.10  on 787  degrees of freedom
## Residual deviance:  778.24  on 780  degrees of freedom
## AIC: 794.24
##
## Number of Fisher Scoring iterations: 5
```

Findings and comparrison between models

- **Analysis——-rent_glm_full——rent_glm_back**

Fisher iterations——model converges——model converge

AIC——-797.69——794.24

Residual Deviance——777.69——778.24

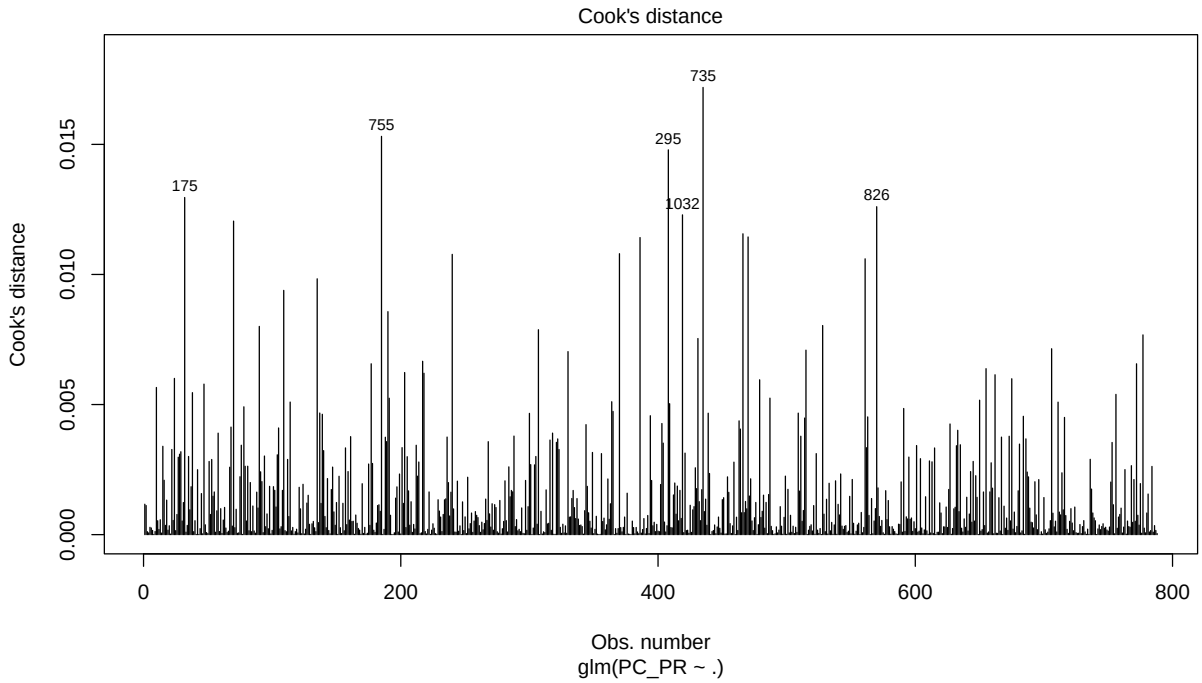
z-values——6/9——6/7

Parameter Co-Efficients——6/8——5/6

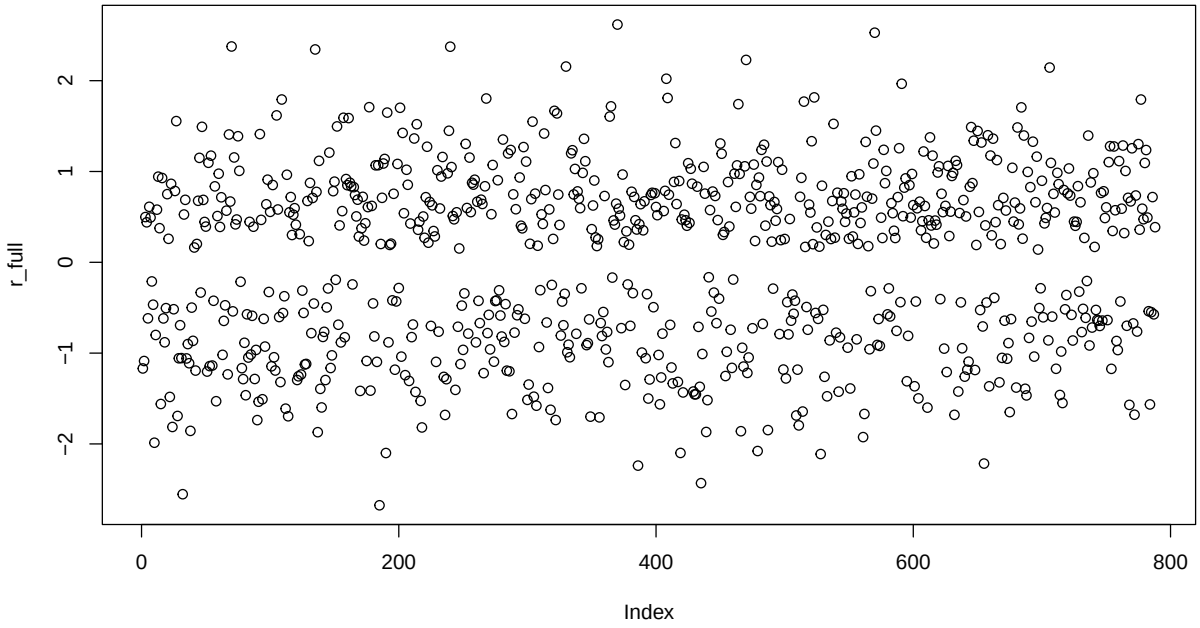
- Both models have converged, meaning that those have found an optimal solution.
- The backward model has a lower AIC, this is because some less significant variables were eliminated. (TotFloor_PR)
- Regarding to Residual Deviance, the full model fits the data a slightly better that the backward model because the back model has removed two variables, so the residuals go up. However, the difference between two models is not significant, both models provide an improvement over the null model.
- In terms of Z Values, the full model indicates that only 6 of 8 variables rejected the null hypotesis that coefficient is 0. But the backward model suggest better results, with 5 of 6 coefficient indicating that backward model is more significant.
- Almost all variables are consistent with correlation matrix. In the full model **TotFloor_PR** and **City_PR** are inconsistent and in the back model only the city showed a different correlation.

3. significantly influential datapoints

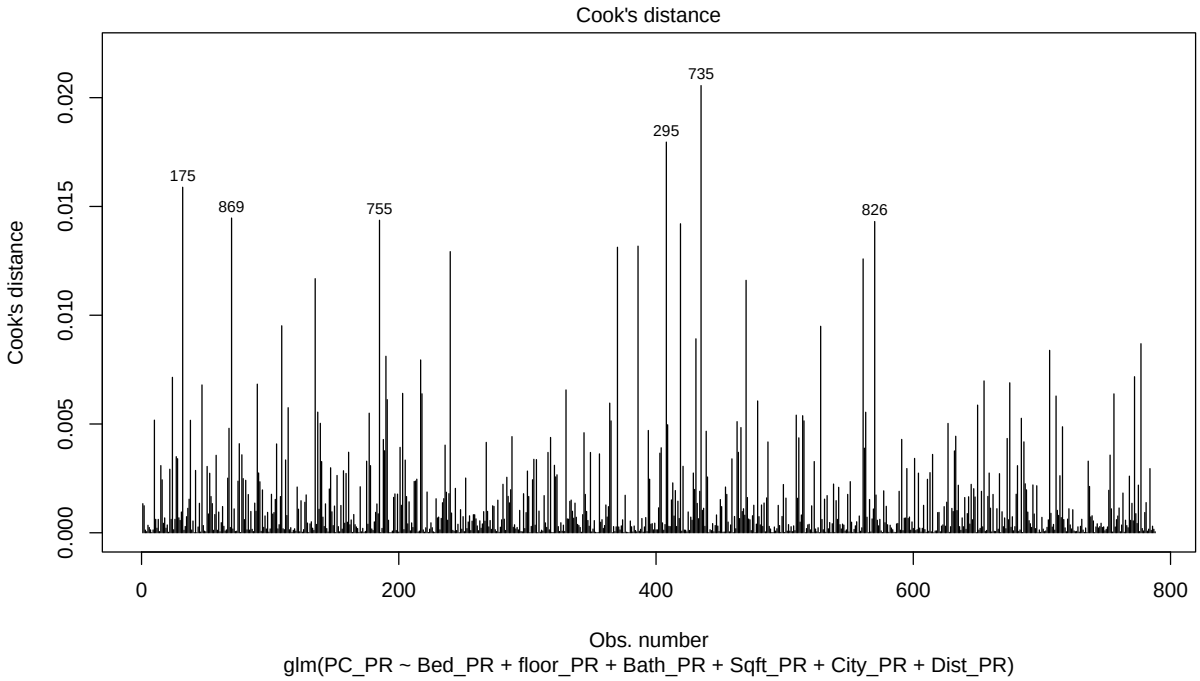
```
# analyze the output for full model
plot(rent_glm_full,which=4, id.n=6)
```



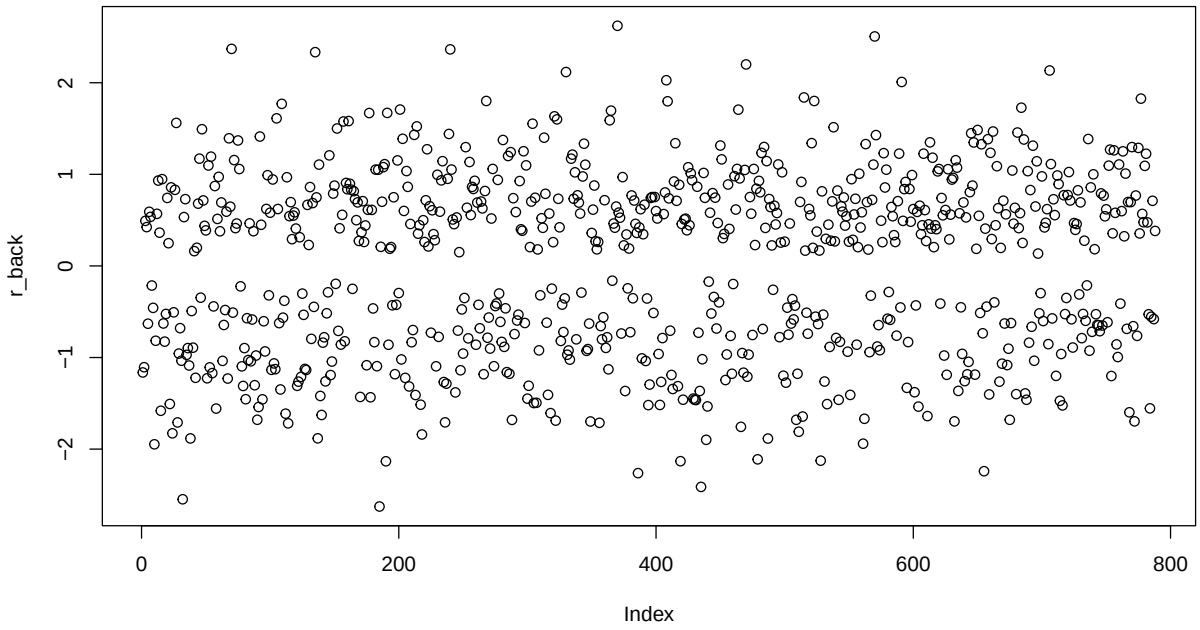

```
#Residuals
r_full <- residuals(rent_glm_full)
plot(r_full)
```




```
# analyze the output for full model
plot(rent_glm_back,which=4, id.n=6)
```



```r
#Residuals
r_back <- residuals(rent_glm_back)
plot(r_back)
```



There are not influential points or observations in either the full model or back model. All points are bellow 0.5. In addition, the scatter plots are not showing any pattern, indicating that residuals have variability around 0.

4.Recommendation

As was explain above, the main measures have no significant differences. However, the backward model eliminated two variables that did not have impact on the model (Comp_PR and TotFloor_PR), making it more efficient due to its simplicity (less variables).

PART B - Fitting models

1. Logistic Regression

In this step I created a new model, this time with direction parameter in "both". The result was the same than the backward because we start from the full model. Also, I calculated the processing time and confusion matrix for future comparisons.

```r
# Creating probabilities with times
start_time <- Sys.time()
rent_glm_stp <- step(rent_glm_full, direction="both", trace=FALSE)
end_time <- Sys.time()
# Calculate the time (in seconds)
```

```
t_glm_PR <- end_time - start_time

# Using the step option in the glm function to fit the model
resp_glm_PR <- predict(rent_glm_stp, newdata=train_pr, type="response")
# Classifying probabilites
class_glm <- ifelse(resp_glm_PR > 0.5, "1","0")
# Creating confussion matrix
Conf_glm_tn <- table(train_pr$PC_PR, class_glm, dnn=list("Actual Rent","Predicted") )

# Printing results
t_glm_PR
```

```
## Time difference of 0.128104 secs
```

```
print("Confusion Matrix with Logistic Regression - Tainning")
```

```
## [1] "Confusion Matrix with Logistic Regression - Tainning"
```

```
Conf_glm_tn
```

```
##            Predicted
## Actual Rent   0   1
##           H 227 114
##           L  80 367
```

Now creating confusion matrix with test dataset

```
# Creating probabilities with test
resp_glm_test_PR <- predict(rent_glm_stp, newdata=test_pr, type="response")
# Classifying probabilites
class_glm_test <- ifelse(resp_glm_test_PR > 0.5, "1","0")
# Creating confussion matrix
Conf_glm_test <- table(test_pr$PC_PR, class_glm_test, dnn=list("Actual Rent","Predicted") )

# Printing results
print("Confusion Matrix with Logistic Regression - Test")
```

```
## [1] "Confusion Matrix with Logistic Regression - Test"
```

```
Conf_glm_test
```

```
##            Predicted
## Actual Rent   0   1
##           H  82  27
##           L  31 123
```

2. Naïve-Bayes Classification

```r
# Creating probabilities with times
start_time <- Sys.time()
rent_nb <- NaiveBayes(PC_PR ~ . , data = train_pr, na.action=na.omit)
end_time <- Sys.time()
# Calculate the time (in seconds)
t_nb_PR <- end_time - start_time

# Using the NaiveBayes model to fit the model
resp_nb_PR <- predict(rent_nb, newdata=train_pr)
# Creating confussion matrix
Conf_nb_tn <- table(Actual=train_pr$PC_PR, Predicted=resp_nb_PR$class)

# Printing results
t_nb_PR
```

```
## Time difference of 0.004257917 secs
```

```r
print("Confusion Matrix with Naïve-Bayes - Tainning")
```

```
## [1] "Confusion Matrix with Naïve-Bayes - Tainning"
```

```r
Conf_nb_tn
```

```
##       Predicted
## Actual   H   L
##      H 213 128
##      L  72 375
```

Now creating confusion matrix with test dataset

```r
# Creating probabilities with test
resp_nb_test_PR <- predict(rent_nb, newdata=test_pr)
# Creating confussion matrix
Conf_nb_test <- table(Actual=test_pr$PC_PR, Predicted=resp_nb_test_PR$class)

# Printing results
print("Confusion Matrix with Naïve-Bayes - Test")
```

```
## [1] "Confusion Matrix with Naïve-Bayes - Test"
```

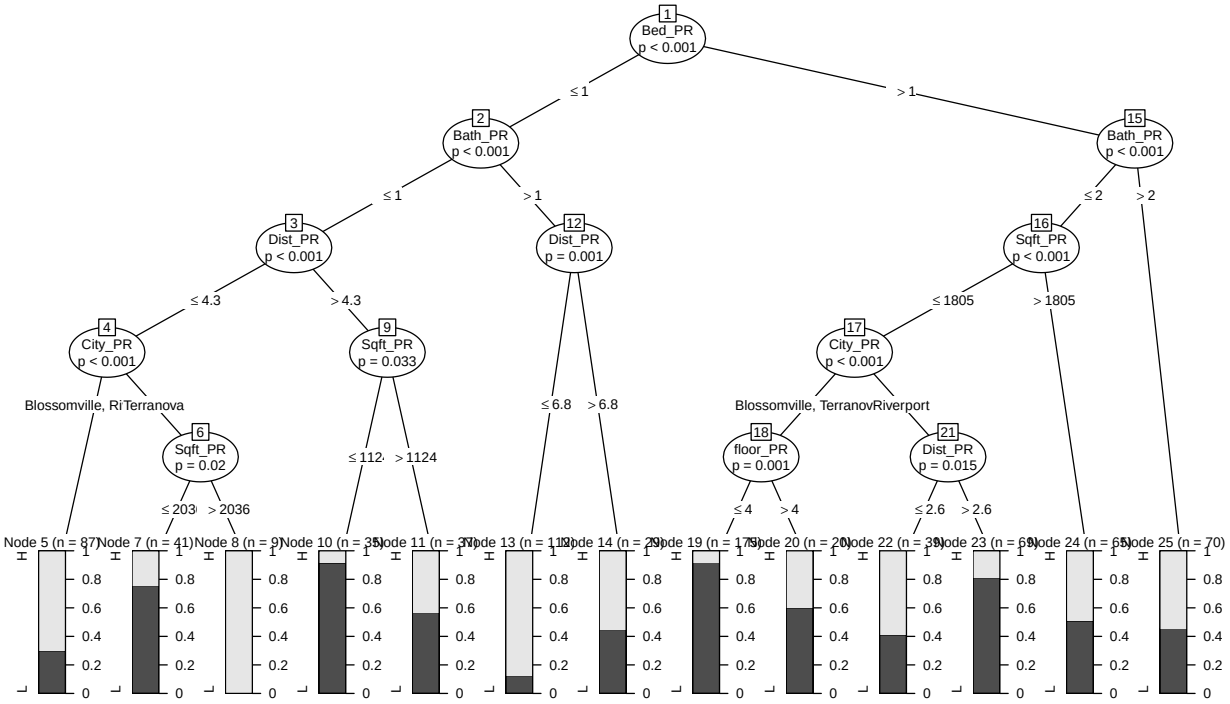
```r
Conf_nb_test
```

```
##       Predicted
## Actual   H   L
##      H  66  43
##      L  31 123
```

3. Recursive Partitioning Analysis

```
# Creating probabilities with times
start_time <- Sys.time()
rent_rp <- ctree(PC_PR ~ . , data = train_pr)
end_time <- Sys.time()
# Calculate the time (in seconds)
t_rp_PR <- end_time - start_time

#plot to analysis
plot(rent_rp, gp=gpar(fontsize=8))
```




```
# Using the recursive partitioning model for fit the model
resp_rp_PR <- predict(rent_rp, newdata=train_pr)
# Creating confussion matrix
Conf_rp_tn <- table(Actual=train_pr$PC_PR, Predicted=resp_rp_PR)

# Printing results
t_rp_PR
```

```
## Time difference of 0.06019092 secs
```

```
print("Confusion Matrix with Recursive Partitioning - Tainning")
```

```
## [1] "Confusion Matrix with Recursive Partitioning - Tainning"
```

```
Conf_rp_tn
```

```
##       Predicted
## Actual   H   L
##      H 245  96
##      L 101 346
```

Now creating confusion matrix with test dataset

```
# Creating probabilities with test
resp_rp_test_PR <- predict(rent_rp, newdata=test_pr)
# Creating confussion matrix
Conf_rp_test <- table(Actual=test_pr$PC_PR, Predicted=resp_rp_test_PR)

# Printing results
print("Confusion Matrix with Recursive Partitioning - Test")
```

```
## [1] "Confusion Matrix with Recursive Partitioning - Test"
```

```
Conf_rp_test
```

```
##       Predicted
## Actual   H   L
##      H  74  35
##      L  41 113
```

4. Neural Network Fitting

```
# Creating probabilities with times
start_time <- Sys.time()
# Using neural network model with size 4 and range 0.0001 as requested
rent_nn <- nnet(PC_PR ~ . , data = train_pr, size=4, rang=0.0001, maxit=1200, trace=TRUE)
```

```
## # weights:  45
## initial  value 546.198308
## iter  10 value 489.123690
## iter  20 value 422.400855
## iter  30 value 393.281970
## iter  40 value 387.006228
## iter  50 value 386.636639
## final  value 386.628807
## converged
```

```
end_time <- Sys.time()
# Calculate the time (in seconds)
t_nn_PR <- end_time - start_time

# Using the Neural Network model for fit the model
resp_nn_PR <- predict(rent_nn, newdata=train_pr, type="class")
# Creating confussion matrix
Conf_nn_tn <- table(Actual=train_pr$PC_PR, Predicted=resp_nn_PR)

# Printing results
t_nn_PR
```

```
## Time difference of 0.03520608 secs
```

```
print("Confusion Matrix with Neural Network - Tainning")
```

```
## [1] "Confusion Matrix with Neural Network - Tainning"
```

```
Conf_nn_tn
```

```
##       Predicted
## Actual   H   L
##      H 236 105
##      L  91 356
```

Now creating confusion matrix with test dataset

```
# Creating probabilities with test
resp_nn_test_PR <- predict(rent_nn, newdata=test_pr, type="class")
# Creating confussion matrix
Conf_nn_test <- table(Actual=test_pr$PC_PR, Predicted=resp_nn_test_PR)

# Printing results
print("Confusion Matrix with Neural Network - Test")
```

```
## [1] "Confusion Matrix with Neural Network - Test"
```

```
Conf_nn_test
```

```
##       Predicted
## Actual   H   L
##      H  85  24
##      L  36 118
```

5. Compare All Classifiers

1. Which classifier is most accurate?

```
# Calculating accuracy on all models and training and testing datasets

#GLM MODEL - TRAIN
TOTAL_GLM_TN <- sum(Conf_glm_tn)
TP_GLM_TN <- Conf_glm_tn[2,2]
TN_GLM_TN <- Conf_glm_tn[1,1]
FP_GLM_TN <- Conf_glm_tn[1,2]
FN_GLM_TN <- Conf_glm_tn[2,1]
#Accuracy
ACC_GLM_TN <- (TP_GLM_TN+TN_GLM_TN)/TOTAL_GLM_TN

print(paste("Accuracy glm model train:", round(ACC_GLM_TN, 3)))
```

```
## [1] "Accuracy glm model train: 0.754"
```

```r
#GLM MODEL - TEST
TOTAL_GLM_TST <- sum(Conf_glm_test)
TP_GLM_TST <- Conf_glm_test[2,2]
TN_GLM_TST <- Conf_glm_test[1,1]
FP_GLM_TST <- Conf_glm_test[1,2]
FN_GLM_TST <- Conf_glm_test[2,1]
#Accuracy
ACC_GLM_TST <- (TP_GLM_TST+TN_GLM_TST)/TOTAL_GLM_TST

print(paste("Accuracy glm model train:", round(ACC_GLM_TST, 3)))
```

```
## [1] "Accuracy glm model train: 0.779"
```

```r
# Naïve-Bayes MODEL - TRAIN
TOTAL_NB_TN <- sum(Conf_nb_tn)
TP_NB_TN <- Conf_nb_tn[2,2]
TN_NB_TN <- Conf_nb_tn[1,1]
FP_NB_TN <- Conf_nb_tn[1,2]
FN_NB_TN <- Conf_nb_tn[2,1]
#Accuracy
ACC_NB_TN <- (TP_NB_TN+TN_NB_TN)/TOTAL_NB_TN

print(paste("Accuracy Naïve-Bayes model train:", round(ACC_NB_TN, 3)))
```

```
## [1] "Accuracy Naïve-Bayes model train: 0.746"
```

```r
# Naïve-Bayes MODEL - TEST
TOTAL_NB_TST <- sum(Conf_nb_test)
TP_NB_TST <- Conf_nb_test[2,2]
TN_NB_TST <- Conf_nb_test[1,1]
FP_NB_TST <- Conf_nb_test[1,2]
FN_NB_TST <- Conf_nb_test[2,1]
#Accuracy
ACC_NB_TST <- (TP_NB_TST+TN_NB_TST)/TOTAL_NB_TST

print(paste("Accuracy Naïve-Bayes model train:", round(ACC_NB_TST, 3)))
```

```
## [1] "Accuracy Naïve-Bayes model train: 0.719"
```

```r
# Recursive Partitioning MODEL - TRAIN
TOTAL_RP_TN <- sum(Conf_rp_tn)
TP_RP_TN <- Conf_rp_tn[2,2]
TN_RP_TN <- Conf_rp_tn[1,1]
FP_RP_TN <- Conf_rp_tn[1,2]
FN_RP_TN <- Conf_rp_tn[2,1]
#Accuracy
ACC_RP_TN <- (TP_RP_TN+TN_RP_TN)/TOTAL_RP_TN

print(paste("Accuracy Recursive Partitioning model train:", round(ACC_RP_TN, 3)))
```

```
## [1] "Accuracy Recursive Partitioning model train: 0.75"
```

```r
# Recursive Partitioning MODEL - TEST
TOTAL_RP_TST <- sum(Conf_rp_test)
TP_RP_TST <- Conf_rp_test[2,2]
TN_RP_TST <- Conf_rp_test[1,1]
FP_RP_TST <- Conf_rp_test[1,2]
FN_RP_TST <- Conf_rp_test[2,1]
#Accuracy
ACC_RP_TST <- (TP_RP_TST+TN_RP_TST)/TOTAL_RP_TST

print(paste("Accuracy Recursive Partitioning model train:", round(ACC_RP_TST, 3)))
```

```
## [1] "Accuracy Recursive Partitioning model train: 0.711"
```

```r
# Neural Network MODEL - TRAIN
TOTAL_NN_TN <- sum(Conf_nn_tn)
TP_NN_TN <- Conf_nn_tn[2,2]
TN_NN_TN <- Conf_nn_tn[1,1]
FP_NN_TN <- Conf_nn_tn[1,2]
FN_NN_TN <- Conf_nn_tn[2,1]
#Accuracy
ACC_NN_TN <- (TP_NN_TN+TN_NN_TN)/TOTAL_NN_TN

print(paste("Accuracy Neural Network model train:", round(ACC_NN_TN, 3)))
```

```
## [1] "Accuracy Neural Network model train: 0.751"
```

```r
# Neural Network MODEL MODEL - TEST
TOTAL_NN_TST <- sum(Conf_nn_test)
TP_NN_TST <- Conf_nn_test[2,2]
TN_NN_TST <- Conf_nn_test[1,1]
FP_NN_TST <- Conf_nn_test[1,2]
FN_NN_TST <- Conf_nn_test[2,1]
#Accuracy
ACC_NN_TST <- (TP_NN_TST+TN_NN_TST)/TOTAL_NN_TST

print(paste("Accuracy Neural Network model train:", round(ACC_NN_TST, 3)))
```

```
## [1] "Accuracy Neural Network model train: 0.772"
```

The most accurate model on both trainning and testing datasets is the glm model with 75.4% in train and 77.9% in test.

The model with the lowest accuracy in the training dataset is Naive-Bayes with 74.6% and in the test dataset it is Recursive Partitioning with 71/1%

2. Which classifier seems most consistent?

To evaluate this, I created a metric to compare the differences between the training and test accuracy metrics for each model.

```r
#GLM MODEL DIFF
DIFF_GLM <- ACC_GLM_TN - ACC_GLM_TST

print(paste("Difference glm model train vs test:", round(DIFF_GLM, 3)))
```

```
## [1] "Difference glm model train vs test: -0.026"
```

```r
# Naïve-Bayes MODEL DIFF
DIFF_NB <- ACC_NB_TN - ACC_NB_TST

print(paste("Difference Naïve-Bayes model train vs test:", round(DIFF_NB, 3)))
```

```
## [1] "Difference Naïve-Bayes model train vs test: 0.028"
```

```r
# Recursive Partitioning MODEL DIFF
DIFF_RP <- ACC_RP_TN - ACC_RP_TST

print(paste("Difference Recursive Partitioning train vs test:", round(DIFF_RP, 3)))
```

```
## [1] "Difference Recursive Partitioning train vs test: 0.039"
```

```r
# Neural Network MODEL DIFF
DIFF_NN <- ACC_NN_TN - ACC_NN_TST

print(paste("Difference Neural Network train vs test:", round(DIFF_NN, 3)))
```

```
## [1] "Difference Neural Network train vs test: -0.021"
```

The recursive partitioning model is the most overfitted because the difference between training and testing is the highest with 0.039 difference.

Neural Network and Logistic Regression models have the best balance between training and testing, which means that those are well-fitting models.

3. Which classifier is most suitable when processing speed is most important?

I will compare all the times

```r
# GLM
t_glm_PR
```

```
## Time difference of 0.128104 secs
```

```r
# Naïve-Bayes
t_nb_PR
```

```
## Time difference of 0.004257917 secs
```

```
# Recursive Partitioning
t_rp_PR
```

```
## Time difference of 0.06019092 secs
```

```
# Neural Network
t_nn_PR
```

```
## Time difference of 0.03520608 secs
```

The best performing model is Naïve-Bayes, the processing time was shorter with 0.02492404

4. Which classifier minimizes false positives?

Retrieving the FP from all models:

```
# GLM train
FP_GLM_TN
```

```
## [1] 114
```

```
# GLM test
FP_GLM_TST
```

```
## [1] 27
```

```
# Naïve-Bayes train
FP_NB_TN
```

```
## [1] 128
```

```
# Naïve-Bayes test
FP_NB_TST
```

```
## [1] 43
```

```
# Recursive Partitioning train
FP_RP_TN
```

```
## [1] 96
```

```
# Recursive Partitioning test
FP_RP_TST
```

```
## [1] 35
```

```
# Neural Network train
FP_NN_TN
```

```
## [1] 105
```

```
# Neural Network test
FP_NN_TST
```

```
## [1] 24
```

Based on the results, the Neural Network have less false positive in test, this makes it the best model that minimizes false positives.

Although recursive partitioning model has less FP in training, the difference is not significant with the NN model. I think that is more important to evaluate the results in the test dataset for this question.

5. In your opinion, which classifier is best overall? Make sure you state why.

Calculating specificity TN/(TN+FP) and sensibility TP/(TP+FN) for more evidence

```
# GLM specificity train
TN_GLM_TN/(TN_GLM_TN+FP_GLM_TN)
```

```
## [1] 0.6656891
```

```
# GLM specificity test
TN_GLM_TST/(TN_GLM_TST+FP_GLM_TST)
```

```
## [1] 0.7522936
```

```
# GLM Sensitivity train
TP_GLM_TN/(TP_GLM_TN+FN_GLM_TN)
```

```
## [1] 0.8210291
```

```
# GLM Sensitivity test
TP_GLM_TST/(TP_GLM_TST+FN_GLM_TST)
```

```
## [1] 0.7987013
```

```
# Naïve-Bayes specificity train
TN_NB_TN/(TN_NB_TN+FP_NB_TN)
```

```
## [1] 0.6246334
```

```
# Naïve-Bayes specificity test
TN_NB_TST/(TN_NB_TST+FP_NB_TST)
```

```
## [1] 0.6055046
```

```
# Naïve-Bayes Sensitivity train
TP_NB_TN/(TP_NB_TN+FN_NB_TN)
```

```
## [1] 0.8389262
```

```
# Naïve-Bayes Sensitivity test
TP_NB_TST/(TP_NB_TST+FN_NB_TST)
```

```
## [1] 0.7987013
```

```
# Recursive Partitioning specificity train
TN_RP_TN/(TN_RP_TN+FP_RP_TN)
```

```
## [1] 0.7184751
```

```
# Recursive Partitioning specificity test
TN_RP_TST/(TN_RP_TST+FP_RP_TST)
```

```
## [1] 0.6788991
```

```
# Recursive Partitioning Sensitivity train
TP_RP_TN/(TP_RP_TN+FN_RP_TN)
```

```
## [1] 0.7740492
```

```
# Recursive Partitioning Sensitivity test
TP_RP_TST/(TP_RP_TST+FN_RP_TST)
```

```
## [1] 0.7337662
```

```
# Neural Network specificity train
TN_NN_TN/(TN_NN_TN+FP_NN_TN)
```

```
## [1] 0.6920821
```

```
# Neural Network specificity test
TN_NN_TST/(TN_NN_TST+FP_NN_TST)
```

```
## [1] 0.7798165
```

```
# Neural Network Sensitivity train
TP_NN_TN/(TP_NN_TN+FN_NN_TN)
```

```
## [1] 0.7964206
```

```r
# Neural Network Sensitivity test
TP_NN_TST/(TP_NN_TST+FN_NN_TST)
```

```
## [1] 0.7662338
```

The best answer to this question is: It depends.

If we want the most accurate model, the LGM model would undoubtedly be the best option, as it has the highest percentage of accuracy and is the best model in terms of specificity in tests. However, this model has a longer processing time. This is why, if time is the priority, we could opt for another model, the NB model.

If, on the other hand, our problem is false positives, the model that favors us is the NN with only 24 FP.

So, depending on the model selection, it depends on what we are looking for.

References

Conestoga College. (2024). PROG8435 – Data Analysis, Modeling and Algorithms - LECTURE 10 – CLASSIFICATION TECHNIQUES (1) LOGISTIC REGRESSION [PowerPoint slides]. eConestoga. Conestoga College. (2024). PROG8435 – Data Analysis, Modeling and Algorithms - LECTURE 11 – CLASSIFICATION TECHNIQUES (2) LOGISTIC REGRESSION [PowerPoint slides]. eConestoga. Conestoga College. (2024). PROG8435 – Data Analysis, Modeling and Algorithms - LECTURE 12 – CLASSIFICATION TECHNIQUES (3) NAIVE-BAYES, NEURAL NETWORKS, RECURSIVE PARTITIOING [PowerPoint slides]. eConestoga. cat function - RDocumentation. (n.d.). https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/cat