Supervised exam

Matteo Ciarrocchi

Abstract

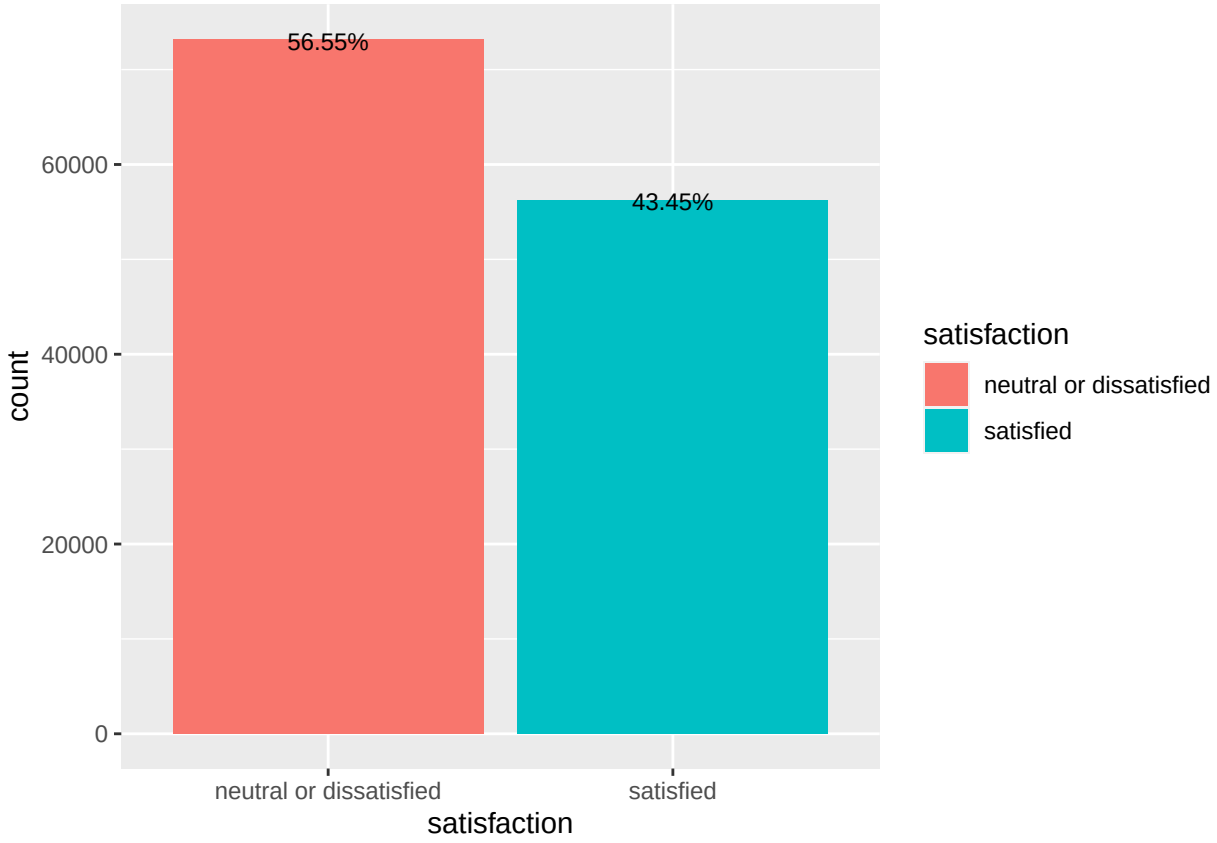
Measuring customer satisfaction is a key element for businesses nowadays because it can significantly contribute to a continuing effort in order to improve service quality. In order to meet customer expectations and achieve higher quality levels, airlines need to understand which are the drivers of passenger satisfaction. Having as a topic of interest customers' satisfaction,this dataset is used for a classification problem, using different techniques: logistic regression, tree predictors and k-NN algorithms. I have found great predictive performances, around 93%, and an easy interpretation of data that highlighted the online boarding and the inflight wifi service as key features for the satisfaction.

Problem understanding and data description

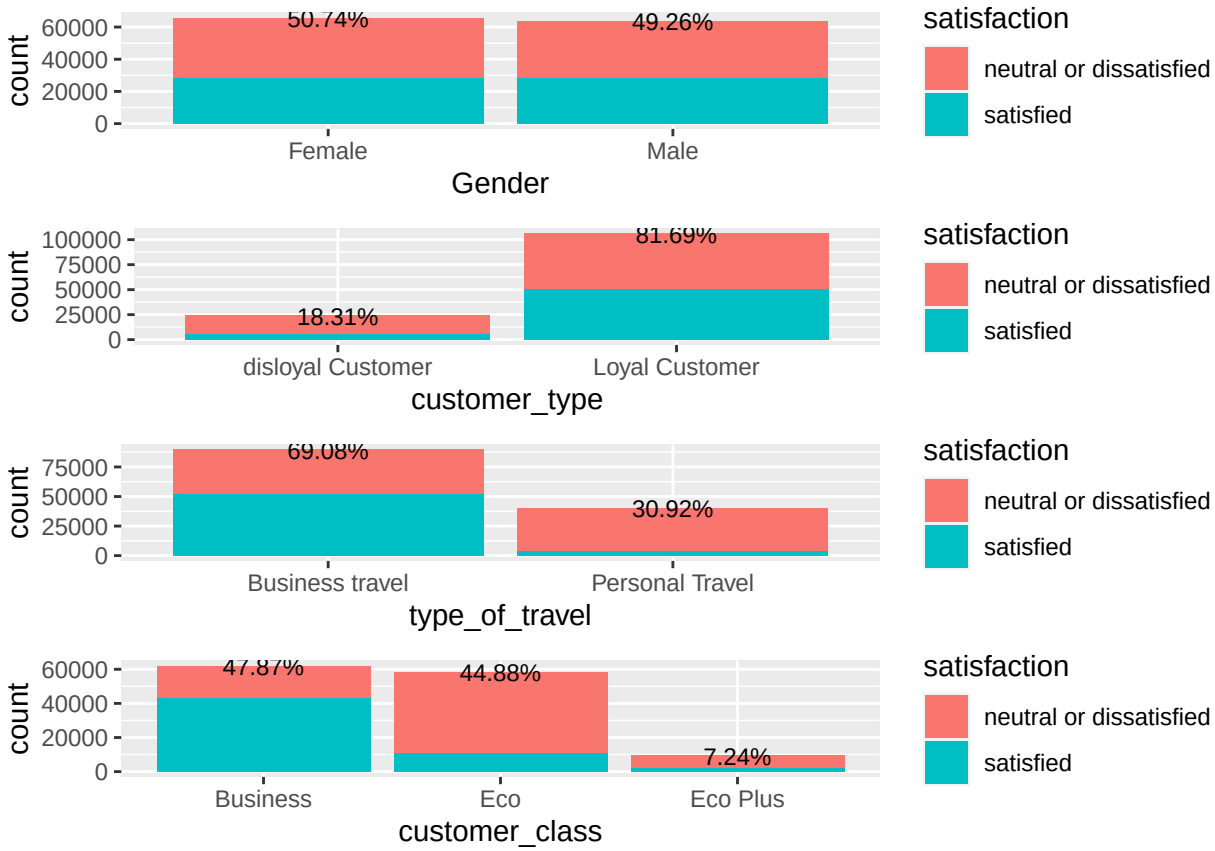
The level of competition in the airline market has been reaching high levels. The fields of battle of the firms are various: some of them aim to shrink the ticket price and to struggle on the price, others try to set a high-level of services and to become a quality leader. In this setting, the satisfaction of the customers is the key to survival. The dataset which I used comes from kaggle, it contains almost 130000 observations and it aims to analyze the discrete variable **satisfaction** (*satisfied* and *neutral or not satisfied*) depending on a set of regressors. After a slight data manipulation made to drop irrelevant data, I performed my analysis with the resulting variables presented below:

- **satisfaction** = the dependent variable, dichotomous variable, *satisfied* or *neutral or dissatisfied*.
- **gender** = dichotomous variable, *male* or *female*.
- **customer_type** = dichotomous variable, *loyal customer* or *disloyal customer*.
- **age** = continuous variable.
- **type_of_travel** = dichotomous variable, *business travel* or *personal travel*.
- **customer class** = dichotomous variable, *business* or *eco*.
- **flight_distance** = continuous variable.
- **inflight_wifi_service** = discrete variable, assigned score between 0 and 5.
- **department_arrival_time_convenient** = discrete variable, assigned score between 0 and 5.
- **ease_of_online_booking** = discrete variable, assigned score between 0 and 5.
- **gate_location** = discrete variable, assigned score between 0 and 5.
- **food_and_drink** = discrete variable, assigned score between 0 and 5.
- **online_boarding** = discrete variable, assigned score between 0 and 5.
- **seat_comfort** = discrete variable, assigned score between 0 and 5.
- **inflight_entertainment** = discrete variable, assigned score between 0 and 5.
- **onboard_service** = discrete variable, assigned score between 0 and 5.
- **leg_room_service** = discrete variable, assigned score between 0 and 5.
- **baggage_handling** = discrete variable, assigned score between 0 and 5.
- **checkin_service** = discrete variable, assigned score between 0 and 5.
- **inflight_service** = discrete variable, assigned score between 0 and 5.
- **cleanliness** = discrete variable, assigned score between 0 and 5.
- **departure_delay_in_minutes** = continuous variable, in minutes.
- **arrival_delay_in_minutes** = continuous variable, in minutes.

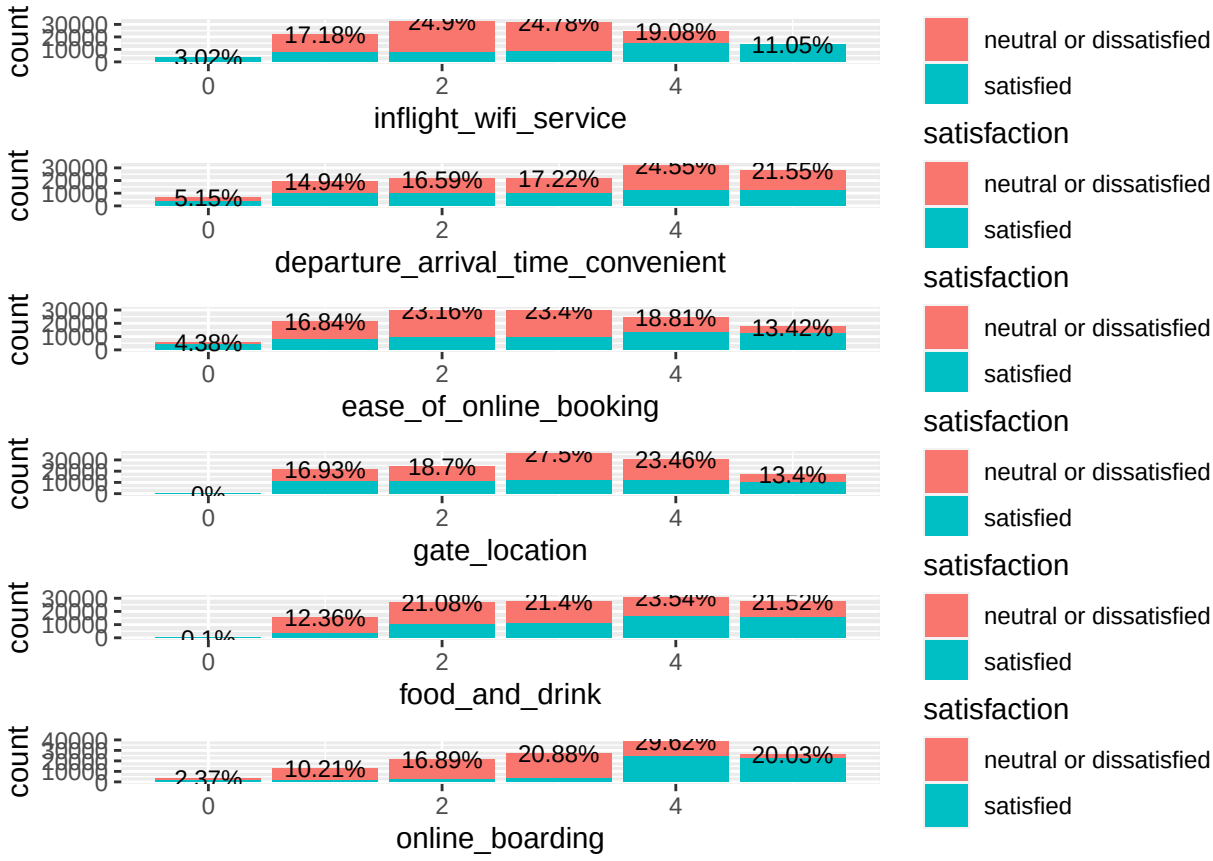
First of all, the histogram of the dependent variable, *satisfaction*, is reported.

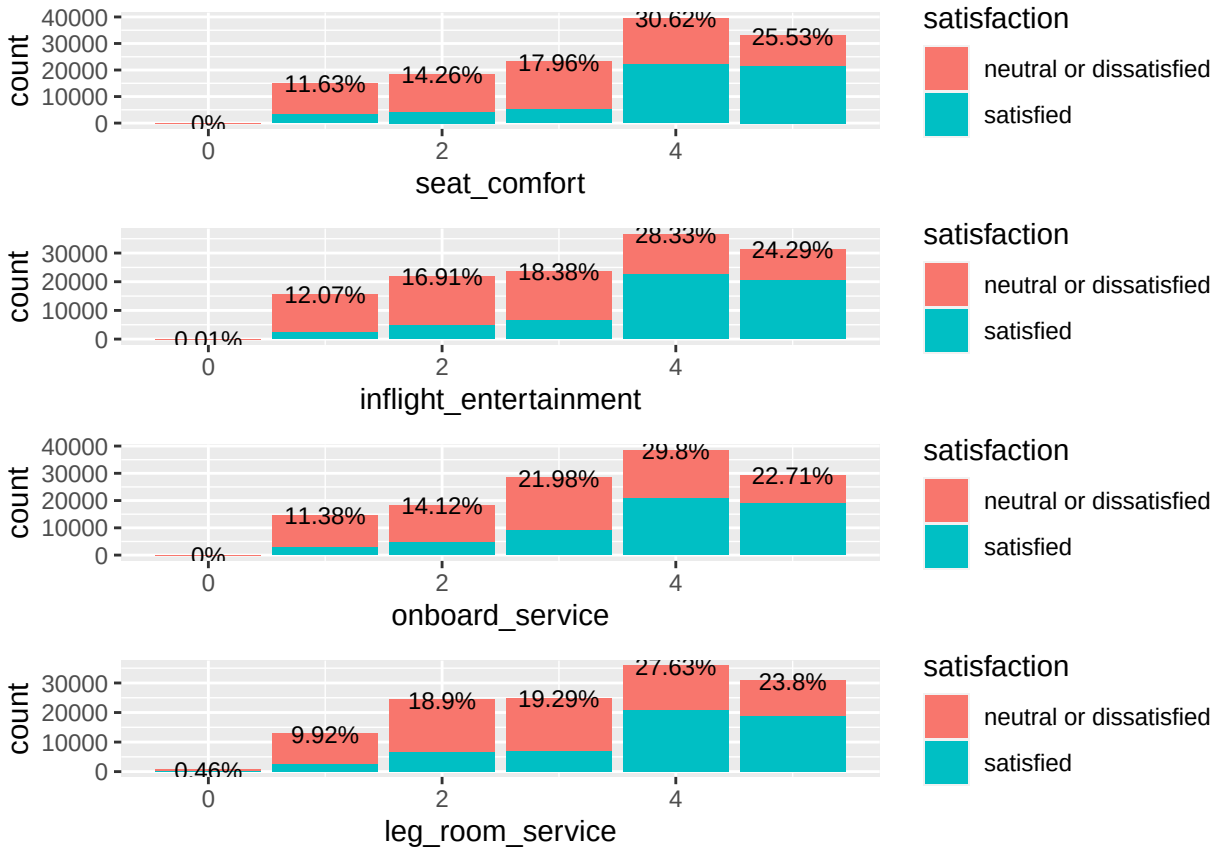


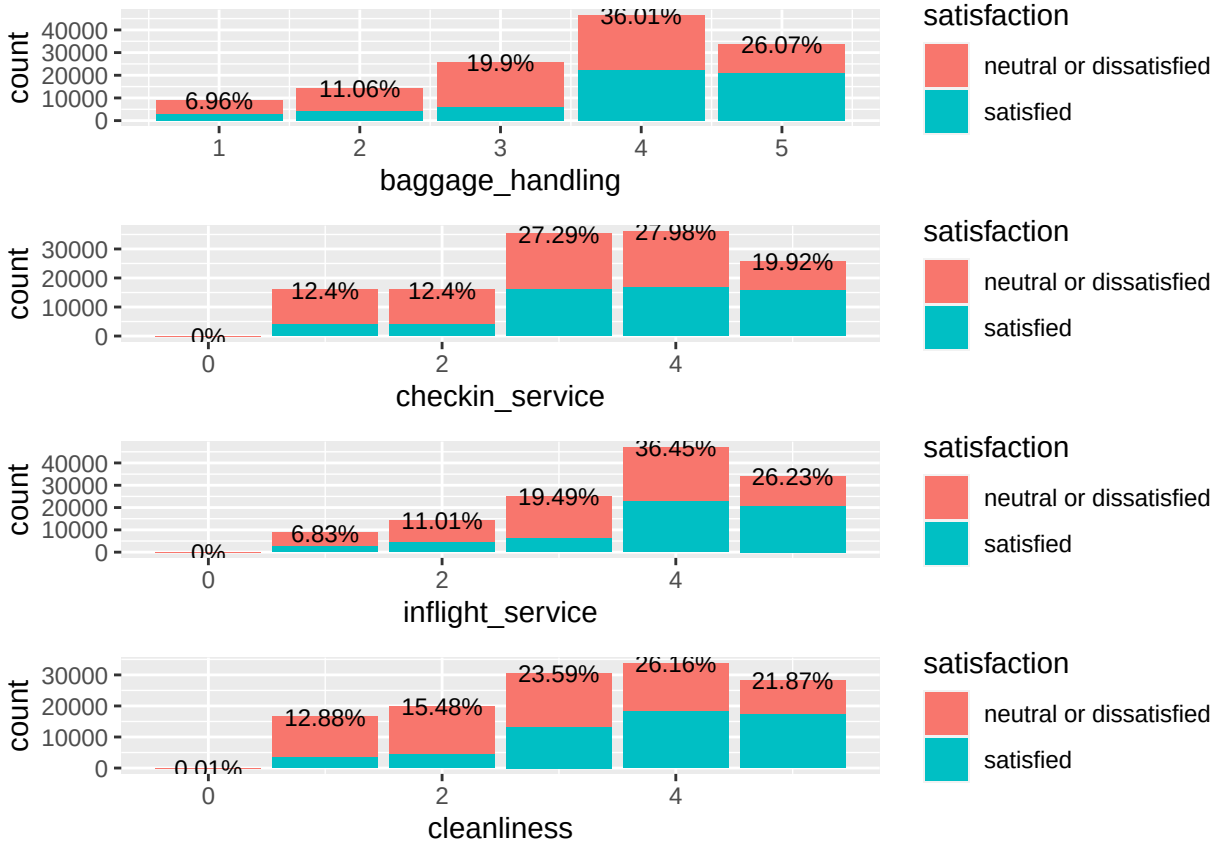
The variable of interest is distributed in a quite balanced way, with *neutral or dissatisfied* customers which are slightly more than *satisfied* ones (56.55% against 43.45%). In the following graph, some categorical variables related to the passengers and their travel's features are presented and the bars are filled in with the rate of satisfaction, just to have an idea whether a category could affect satisfaction or not and could have a significant coefficient in the logistic regression of the next section.



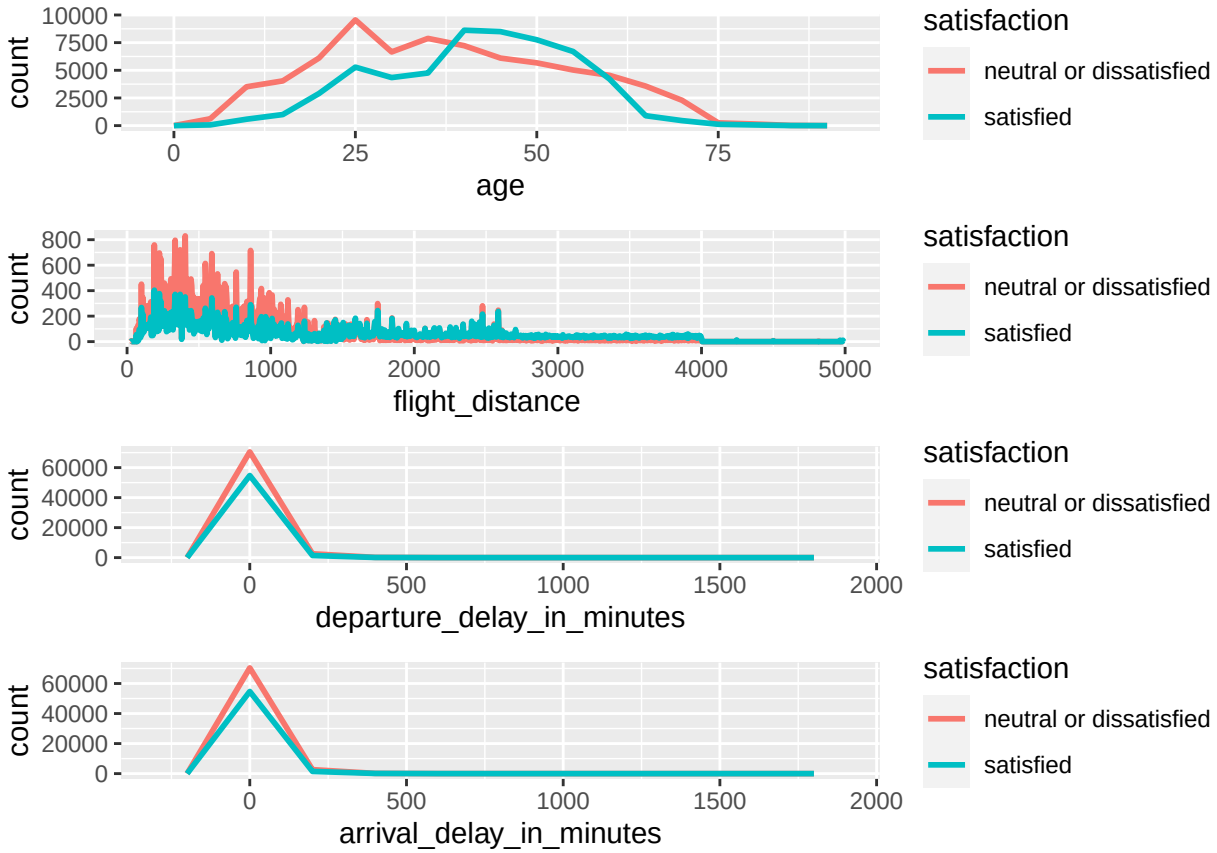
As it can be foreseeable, higher percentage of satisfaction are present in the *loyal customer* category rather than in the *disloyal customer* one and customers are more satisfied by *business* travels rather than the personal ones. *Customer-class* also seems relevant with greater satisfaction in the *business class*, while there is an equal distribution in the *gender* category. The results for the other categorical variables, evaluated by the customers with a grade between 0 and 5 about the fly experience, are shown below.


count
inflight_wifi_service
3.02%
17.18%
24.9%
24.78%
19.08%
11.05%
departure_arrival_time_convenient
5.15%
14.94%
16.59%
17.22%
24.55%
21.55%
ease_of_online_booking
4.38%
16.84%
23.16%
23.4%
18.81%
13.42%
gate_location
16.93%
18.7%
27.5%
23.46%
13.4%
food_and_drink
0.1%
12.36%
21.08%
21.4%
23.54%
21.52%
online_boarding
2.37%
10.21%
16.89%
20.88%
29.62%
20.03%
satisfaction
neutral or dissatisfied
satisfied


count
40000
30000
20000
10000
0
0%
11.63%
14.26%
17.96%
30.62%
25.53%
0
2
4
seat_comfort
satisfaction
neutral or dissatisfied
satisfied
count
30000
20000
10000
0
0.01%
12.07%
16.91%
18.38%
28.33%
24.29%
0
2
4
inflight_entertainment
satisfaction
neutral or dissatisfied
satisfied
count
40000
30000
20000
10000
0
0%
11.38%
14.12%
21.98%
29.8%
22.71%
0
2
4
onboard_service
satisfaction
neutral or dissatisfied
satisfied
count
30000
20000
10000
0
0.46%
9.92%
18.9%
19.29%
27.63%
23.8%
0
2
4
leg_room_service
satisfaction
neutral or dissatisfied
satisfied



Summarizing the information which can be deduced from the images above, a good rank of the *wifi_service* brings high satisfaction, as it happens for the *online_boarding* and the *ease_of_online_booking*. The other variables of the first image have not categories which significantly seem to affect *satisfaction*. Looking at the second image, higher ranks within each variable are correlated with higher percentages of satisfaction within the category, but this increase does not seem to be particularly great in any variables. What seems relevant is the jump in the *satisfied* customers percentage passing from rank 3 to 4, especially in *seat_comfort*, *inflight_entertainment* and *leg_room_service*. The description of the continuous variables present in the dataset is reported below.



As the plots show, satisfaction lines about *departure and arrival delays* move together along the different values. The number of satisfied persons respect to the *age* seems to be greater for middle-aged people and smaller for the other ones, while there is a greater number of unsatisfied passengers on the shorter distances (smaller than 1200), but this difference disappears after this threshold.

Logistic regression

The starting point of this analysis is the logistic regression. First of all, I presented the full model, reported below. I decided to split the dataset in the training and test set with a training size equal to 0.7 respect to the starting dataset. Setting the seed, this is my first result:

```
##
## Call:
## glm(formula = satisfaction ~ ., family = binomial(link = "logit"),
##     data = airlines_clean)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.6576  -0.2140  -0.0470   0.1343   4.4072
##
## Coefficients: (2 not defined because of singularities)
##                                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)                       7.403e+00  2.827e+03   0.003 0.997911
## GenderMale                        6.782e-02  2.442e-02   2.777 0.005486 **
## customer_typeLoyal Customer       3.341e+00  4.411e-02  75.746  < 2e-16 ***
```

```
## age                                 -2.357e-03  9.122e-04  -2.584 0.009758 **
## type_of_travelPersonal Travel       -4.293e+00  4.923e-02 -87.195  < 2e-16 ***
## customer_classEco                   -6.085e-01  3.326e-02 -18.297  < 2e-16 ***
## customer_classEco Plus              -8.368e-01  5.419e-02 -15.443  < 2e-16 ***
## flight_distance                      6.494e-06  1.373e-05   0.473 0.636242
## inflight_wifi_service1              -2.373e+01  8.025e+01  -0.296 0.767495
## inflight_wifi_service2              -2.398e+01  8.025e+01  -0.299 0.765046
## inflight_wifi_service3              -2.401e+01  8.025e+01  -0.299 0.764809
## inflight_wifi_service4              -2.243e+01  8.025e+01  -0.280 0.779855
## inflight_wifi_service5              -1.698e+01  8.025e+01  -0.212 0.832476
## departure_arrival_time_convenient1   3.612e-01  8.312e-02   4.346 1.39e-05 ***
## departure_arrival_time_convenient2   4.571e-01  7.972e-02   5.734 9.81e-09 ***
## departure_arrival_time_convenient3   2.913e-01  7.690e-02   3.788 0.000152 ***
## departure_arrival_time_convenient4 -6.609e-01  6.901e-02  -9.576  < 2e-16 ***
## departure_arrival_time_convenient5 -8.749e-01  7.567e-02 -11.563  < 2e-16 ***
## ease_of_online_booking1              2.963e+00  8.164e-01   3.629 0.000285 ***
## ease_of_online_booking2              2.951e+00  8.164e-01   3.615 0.000300 ***
## ease_of_online_booking3              3.408e+00  8.162e-01   4.175 2.97e-05 ***
## ease_of_online_booking4              4.239e+00  8.160e-01   5.195 2.05e-07 ***
## ease_of_online_booking5              3.606e+00  8.162e-01   4.418 9.97e-06 ***
## gate_location2                       4.894e-02  5.451e-02   0.898 0.369302
## gate_location3                      -1.419e-01  5.036e-02  -2.817 0.004849 **
## gate_location4                      -3.752e-01  5.211e-02  -7.199 6.05e-13 ***
## gate_location5                      -5.988e-01  6.728e-02  -8.900  < 2e-16 ***
## food_and_drink1                     -9.974e-01  1.316e+00  -0.758 0.448611
## food_and_drink2                     -7.048e-01  1.316e+00  -0.536 0.592297
## food_and_drink3                     -8.374e-01  1.316e+00  -0.636 0.524595
## food_and_drink4                     -7.976e-01  1.316e+00  -0.606 0.544482
## food_and_drink5                     -9.759e-01  1.316e+00  -0.742 0.458371
## online_boarding1                    -3.695e+00  8.217e-01  -4.497 6.89e-06 ***
## online_boarding2                    -3.674e+00  8.216e-01  -4.472 7.74e-06 ***
## online_boarding3                    -3.872e+00  8.213e-01  -4.714 2.43e-06 ***
## online_boarding4                    -2.231e+00  8.209e-01  -2.717 0.006579 **
## online_boarding5                    -9.601e-01  8.211e-01  -1.169 0.242287
## seat_comfort2                       -5.006e-01  6.455e-02  -7.755 8.84e-15 ***
## seat_comfort3                       -1.591e+00  6.018e-02 -26.437  < 2e-16 ***
## seat_comfort4                       -9.001e-01  5.959e-02 -15.103  < 2e-16 ***
## seat_comfort5                       -6.655e-02  6.442e-02  -1.033 0.301575
## inflight_entertainment1              4.007e+01  1.429e+03   0.028 0.977624
## inflight_entertainment2              4.082e+01  1.429e+03   0.029 0.977208
## inflight_entertainment3              4.165e+01  1.429e+03   0.029 0.976743
## inflight_entertainment4              4.131e+01  1.429e+03   0.029 0.976935
## inflight_entertainment5              4.054e+01  1.429e+03   0.028 0.977361
## onboard_service1                    -2.400e+01  3.167e+03  -0.008 0.993952
## onboard_service2                    -2.388e+01  3.167e+03  -0.008 0.993983
## onboard_service3                    -2.336e+01  3.167e+03  -0.007 0.994113
## onboard_service4                    -2.328e+01  3.167e+03  -0.007 0.994134
## onboard_service5                    -2.273e+01  3.167e+03  -0.007 0.994274
## leg_room_service1                   -2.480e+00  8.530e-01  -2.907 0.003647 **
## leg_room_service2                   -2.197e+00  8.526e-01  -2.577 0.009960 **
## leg_room_service3                   -2.322e+00  8.525e-01  -2.724 0.006454 **
## leg_room_service4                   -1.629e+00  8.525e-01  -1.911 0.056000 .
## leg_room_service5                   -1.454e+00  8.525e-01  -1.706 0.087967 .
## baggage_handling2                   -2.150e-01  6.823e-02  -3.151 0.001626 **
```

```
## baggage_handling3                 -8.163e-01  6.348e-02 -12.860  < 2e-16 ***
## baggage_handling4                 -1.991e-01  6.165e-02  -3.230 0.001238 **
## baggage_handling5                  4.959e-01  6.562e-02   7.558 4.11e-14 ***
## checkin_service2                   1.763e-01  4.719e-02   3.737 0.000186 ***
## checkin_service3                   7.203e-01  4.205e-02  17.129  < 2e-16 ***
## checkin_service4                   6.915e-01  4.191e-02  16.500  < 2e-16 ***
## checkin_service5                   1.467e+00  4.852e-02  30.236  < 2e-16 ***
## inflight_service1                 -4.816e-01  6.852e-02  -7.028 2.09e-12 ***
## inflight_service2                 -7.091e-01  6.226e-02 -11.389  < 2e-16 ***
## inflight_service3                 -1.406e+00  5.123e-02 -27.445  < 2e-16 ***
## inflight_service4                 -7.123e-01  4.013e-02 -17.752  < 2e-16 ***
## inflight_service5                         NA         NA      NA       NA
## cleanliness1                      -9.835e-01  6.696e-02 -14.688  < 2e-16 ***
## cleanliness2                      -9.945e-01  6.527e-02 -15.237  < 2e-16 ***
## cleanliness3                      -4.852e-01  5.491e-02  -8.836  < 2e-16 ***
## cleanliness4                      -6.192e-01  5.371e-02 -11.529  < 2e-16 ***
## cleanliness5                              NA         NA      NA       NA
## departure_delay_in_minutes         4.521e-03  1.202e-03   3.761 0.000169 ***
## arrival_delay_in_minutes          -8.657e-03  1.189e-03  -7.283 3.25e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 177276  on 129484  degrees of freedom
## Residual deviance:  46213  on 129411  degrees of freedom
## AIC: 46361
##
## Number of Fisher Scoring iterations: 17
```

There are a lot of coefficients in the model since the categorical variables have spanned the number of regressors. Given the results, there are variables which should take values different from 0 to correctly compute the probability of being satisfied of a customer. Assuming different levels of significance between 0.001 and 0.1, *gender*, *age*, *customer_class*, *departure_arrival_time_convenient* are only a few of these variables, which set is identified by the significance codes. Coefficients assume values different from what it was expected, sometimes in a not reasonable way: for example, higher ratings of a feature affect always more negatively the probability of being satisfied. Therefore, I checked the variance inflation factors (vif) of the variables, which give insights into the variables which could cause multicollinearity. Here is reported the result.

```
##
## Attaching package: 'performance'

## The following object is masked from 'package:ROCR':
##
##     performance

## # Check for Multicollinearity
##
## Low Correlation
##
##              Term  VIF Increased SE Tolerance
##            Gender 1.02         1.01      0.98
```

```
##    customer_type 2.00         1.41      0.50
##              age 1.19         1.09      0.84
##   type_of_travel 2.44         1.56      0.41
##   customer_class 1.77         1.33      0.57
##  flight_distance 1.37         1.17      0.73
##  checkin_service 1.35         1.16      0.74
##
## Moderate Correlation
##
##                   Term  VIF Increased SE Tolerance
##  inflight_wifi_service 9.42         3.07      0.11
##           seat_comfort 9.18         3.03      0.11
##        onboard_service 7.19         2.68      0.14
##       baggage_handling 6.51         2.55      0.15
##       inflight_service 9.57         3.09      0.10
##
## High Correlation
##
##                               Term     VIF Increased SE Tolerance
##  departure_arrival_time_convenient   31.69         5.63      0.03
##             ease_of_online_booking 1334.29        36.53      0.00
##                       gate_location   11.82         3.44      0.08
##                     food_and_drink   19.06         4.37      0.05
##                    online_boarding   95.20         9.76      0.01
##             inflight_entertainment  125.38        11.20      0.01
##                   leg_room_service   24.96         5.00      0.04
##                        cleanliness   16.39         4.05      0.06
##         departure_delay_in_minutes   15.48         3.93      0.06
##           arrival_delay_in_minutes   15.52         3.94      0.06
```

There are a lot of high variables with an high correlation, especially the *ease_of_online_booking*. There are also NA values in the logistic regression, sign of presence of linear combination of variables. Correlations are classified considering as threshold the value of vif equal to 10. Hence, I decided to perform a subset selection of my dataset with a forward stepwise approach, discarding at each step the variable with the highest vif until the "high correlation" set was not empty. At the end, the process excluded variables *ease_of_online_booking*, *inflight_entertainment*, *departure_delay_in_minutes* and *departure_arrival_time_convenient*, because their information is carried out by the other regressors. After that, I checked the best possible subset of variables, using the forward stepwise method and the Mallow's Cp, but the results selected all the variables as best set excluding only the NA value of *inflight_service5*. Therefore, the logistic regression model without the variables reported above is reported.

```
##
## Call:
## glm(formula = satisfaction ~ . - ease_of_online_booking - inflight_entertainment -
##     departure_delay_in_minutes - departure_arrival_time_convenient,
##     family = binomial(link = "logit"), data = airlines_clean)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.5192  -0.2390  -0.0552   0.1394   4.1725
##
## Coefficients: (1 not defined because of singularities)
##                                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)                  -3.757e+01  3.201e+03  -0.012 0.990634
```

```
## GenderMale                      8.633e-02  2.403e-02   3.593 0.000327 ***
## customer_typeLoyal Customer     3.231e+00  4.113e-02  78.562  < 2e-16 ***
## age                            -3.710e-03  8.915e-04  -4.161 3.17e-05 ***
## type_of_travelPersonal Travel -4.482e+00  4.547e-02 -98.582  < 2e-16 ***
## customer_classEco             -6.553e-01  3.245e-02 -20.193  < 2e-16 ***
## customer_classEco Plus        -9.177e-01  5.346e-02 -17.166  < 2e-16 ***
## flight_distance                9.621e-06  1.349e-05   0.713 0.475638
## inflight_wifi_service1        -2.335e+01  7.451e+01  -0.313 0.753969
## inflight_wifi_service2        -2.357e+01  7.451e+01  -0.316 0.751810
## inflight_wifi_service3        -2.338e+01  7.451e+01  -0.314 0.753720
## inflight_wifi_service4        -2.173e+01  7.451e+01  -0.292 0.770549
## inflight_wifi_service5        -1.641e+01  7.451e+01  -0.220 0.825665
## gate_location2                 1.029e-01  3.944e-02   2.608 0.009113 **
## gate_location3                -1.365e-02  3.770e-02  -0.362 0.717345
## gate_location4                -3.572e-01  4.032e-02  -8.858  < 2e-16 ***
## gate_location5                -1.090e+00  5.090e-02 -21.412  < 2e-16 ***
## food_and_drink1                1.781e+00  9.828e-01   1.812 0.070034 .
## food_and_drink2                2.279e+00  9.830e-01   2.318 0.020447 *
## food_and_drink3                2.370e+00  9.828e-01   2.412 0.015866 *
## food_and_drink4                2.306e+00  9.828e-01   2.346 0.018954 *
## food_and_drink5                1.961e+00  9.828e-01   1.995 0.046044 *
## online_boarding1              -2.409e-01  1.592e-01  -1.512 0.130423
## online_boarding2              -1.936e-01  1.587e-01  -1.220 0.222537
## online_boarding3              -3.380e-01  1.575e-01  -2.147 0.031825 *
## online_boarding4               1.396e+00  1.558e-01   8.963  < 2e-16 ***
## online_boarding5               2.441e+00  1.582e-01  15.433  < 2e-16 ***
## seat_comfort2                 -3.222e-01  6.110e-02  -5.274 1.34e-07 ***
## seat_comfort3                 -1.305e+00  5.719e-02 -22.818  < 2e-16 ***
## seat_comfort4                 -7.418e-01  5.669e-02 -13.086  < 2e-16 ***
## seat_comfort5                 -2.212e-02  6.049e-02  -0.366 0.714638
## onboard_service1               1.677e+01  2.799e+03   0.006 0.995219
## onboard_service2               1.699e+01  2.799e+03   0.006 0.995158
## onboard_service3               1.766e+01  2.799e+03   0.006 0.994967
## onboard_service4               1.771e+01  2.799e+03   0.006 0.994952
## onboard_service5               1.809e+01  2.799e+03   0.006 0.994843
## leg_room_service1              9.617e-01  2.559e-01   3.758 0.000171 ***
## leg_room_service2              1.302e+00  2.544e-01   5.118 3.08e-07 ***
## leg_room_service3              1.288e+00  2.544e-01   5.064 4.10e-07 ***
## leg_room_service4              2.055e+00  2.538e-01   8.098 5.58e-16 ***
## leg_room_service5              2.105e+00  2.541e-01   8.285  < 2e-16 ***
## baggage_handling2             -9.430e-02  6.630e-02  -1.422 0.154918
## baggage_handling3             -5.891e-01  6.189e-02  -9.518  < 2e-16 ***
## baggage_handling4             -8.375e-02  6.031e-02  -1.389 0.164928
## baggage_handling5              4.804e-01  6.382e-02   7.527 5.19e-14 ***
## checkin_service2               1.668e-01  4.700e-02   3.549 0.000387 ***
## checkin_service3               6.457e-01  4.152e-02  15.552  < 2e-16 ***
## checkin_service4               6.103e-01  4.134e-02  14.765  < 2e-16 ***
## checkin_service5               1.377e+00  4.720e-02  29.162  < 2e-16 ***
## inflight_service1             -5.555e-01  6.543e-02  -8.491  < 2e-16 ***
## inflight_service2             -5.707e-01  5.888e-02  -9.692  < 2e-16 ***
## inflight_service3             -1.086e+00  4.776e-02 -22.738  < 2e-16 ***
## inflight_service4             -5.403e-01  3.756e-02 -14.387  < 2e-16 ***
## inflight_service5                     NA         NA      NA       NA
## cleanliness1                   3.648e+01  1.555e+03   0.023 0.981282
```

```
## cleanliness2                   3.661e+01  1.555e+03   0.024 0.981215
## cleanliness3                   3.731e+01  1.555e+03   0.024 0.980853
## cleanliness4                   3.708e+01  1.555e+03   0.024 0.980974
## cleanliness5                   3.752e+01  1.555e+03   0.024 0.980749
## arrival_delay_in_minutes      -4.603e-03  3.047e-04 -15.107  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 177276  on 129484  degrees of freedom
## Residual deviance:  47979  on 129426  degrees of freedom
## AIC: 48097
##
## Number of Fisher Scoring iterations: 17
```

Now the interpretation of the coefficients seems more reasonable: considering an intercept equal to 0, *loyal customer* variable has a positive and high coefficient, high *online boarding* is strongly positive with high ratings as the *checkin_service* and the *arrival_delay*. After having selected the right number of variables, I decided to split the dataset in training and test sets with a training size equal to 0.7 respect to the starting dataset. Here confusion matrices of, respectively, training and test sets with accuracy, sensitivity and specificity are reported.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    No   Yes
##        No  48699  3372
##        Yes  2558 36011
##
##                Accuracy : 0.9346
##                  95% CI : (0.9329, 0.9362)
##     No Information Rate : 0.5655
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8665
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9144
##             Specificity : 0.9501
##          Pos Pred Value : 0.9337
##          Neg Pred Value : 0.9352
##              Prevalence : 0.4345
##          Detection Rate : 0.3973
##    Detection Prevalence : 0.4255
##       Balanced Accuracy : 0.9322
##
##        'Positive' Class : Yes
##
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction    No   Yes
##        No  20824  1507
##        Yes  1143 15371
##
##                Accuracy : 0.9318
##                  95% CI : (0.9292, 0.9343)
##     No Information Rate : 0.5655
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8608
##
##  Mcnemar's Test P-Value : 1.77e-12
##
##             Sensitivity : 0.9107
##             Specificity : 0.9480
##          Pos Pred Value : 0.9308
##          Neg Pred Value : 0.9325
##              Prevalence : 0.4345
##          Detection Rate : 0.3957
##    Detection Prevalence : 0.4251
##       Balanced Accuracy : 0.9293
##
##        'Positive' Class : Yes
##
```

The results are really good. The accuracy is equal to 0.9346 in the training set and 0.9318 in the test set. Sensitivity, which is the percentage of satisfied customers that are correctly identified, is 0.9144 in the training set and 0.9107 in the test set, while specificity, which is the percentage of neutral or dissatisfied customers that are correctly identified, is respectively equal 0.9501 and 0.9480. Mcnemar's test p-values are clearly smaller than 0.01 and, therefore, marginal frequencies are not equal. In order to add robustness to the results, I ran the model two more times changing the data partition. The findings, reported in the appendix, are consistent with the previous one. In the first model, accuracy has been 0.9333 for the training set and 0.9349 for the test one, sensitivity 0.9118 and 0.9147, specificity 0.9492 and 0.9504. The second model has an accuracy of 0.934 and 0.9327, sensitivity equal to 0.9129 and 0.9125, specificity equal to 0.9502 and 0.9482. All these three results are good and consistent with each other, the accuracy is very great and no overfitting is present since train and test accuracies are really near. I also tried passing the train size from 0.7 to 0.9 of the starting dataset.The result is slightly improved: training and test accuracies are equal to 0.9333 and 0.9369, sensitivity equal to 0.9122 and 0.9175 and specificity equal to 0.9496 and 0.9518. The validation approach with which I estimated test errors is based only on a subset of observations, and this usually implies underestimated accuracy. Therefore, I used a 10-fold cross validation approach to detect possible inconsistency in the data, but the result isn't far away from the previous one: accuracy = 0.9336, very similar with respect to the validation approach.

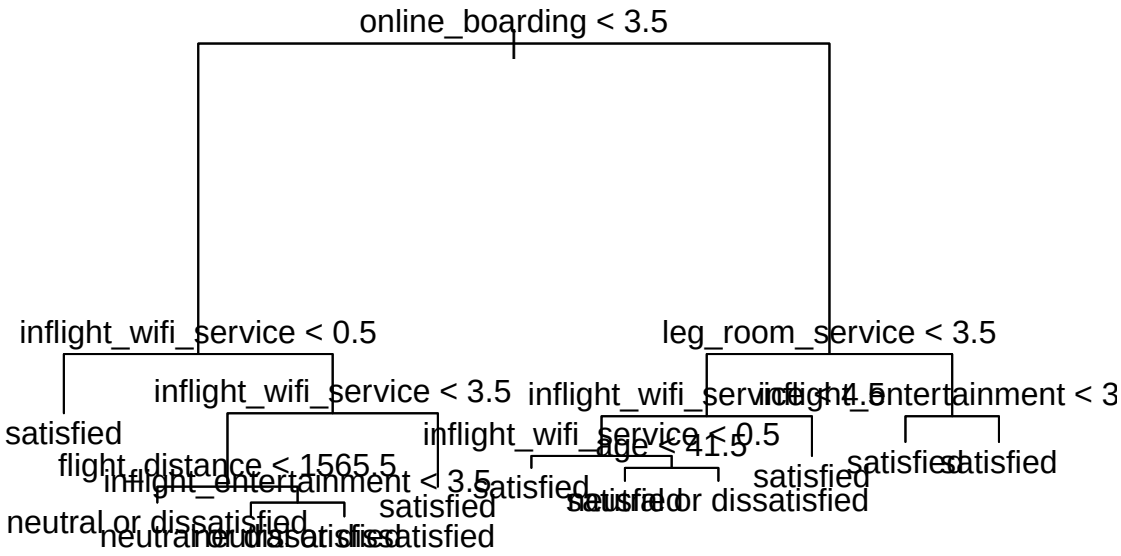
Furthermore, I ran the the logistic regression normalizing the continuous variables (*age*,*flight_distance* and *arrival_delay_in_minutes*) of the training and the test sets to weight the effects of their possible different units of measure. Three regressions are run with two different data partition and a 10-cross validation and the results are really similar to the previous ones: test accuracy respectively equal to 0.9316, 0.9335 and 0.9337, this was expected because continuous variables have similar units of measure (two out of three are minutes) and they are few respect to the categorical ones, therefore they have less weight in the computation. The results are reported in the appendix. To better interpret the puzzling roles of the variables, I decided to use tree classifiers.

Trees

In this section I used tree classifiers to fit my data, since they give great advantages in terms of interpretability. The accuracy results of the previous section were really good. For this reason, I did not expect high-level performances for the prediction accuracy, but only something that is not too bad and that allows for an easy interpretation. I started with a tree including all the variables. The tree plot and a summary containing useful information are reported below.

```
## Registered S3 method overwritten by 'tree':
##   method     from
##   print.tree cli
```

```
##
## Classification tree:
## tree(formula = satisfaction ~ ., data = airlines_clean)
## Variables actually used in tree construction:
## [1] "online_boarding"        "inflight_wifi_service"  "flight_distance"
## [4] "inflight_entertainment" "leg_room_service"       "age"
## Number of terminal nodes:  11
## Residual mean deviance:  0.6497 = 84110 / 129500
## Misclassification error rate: 0.1415 = 18325 / 129486
```



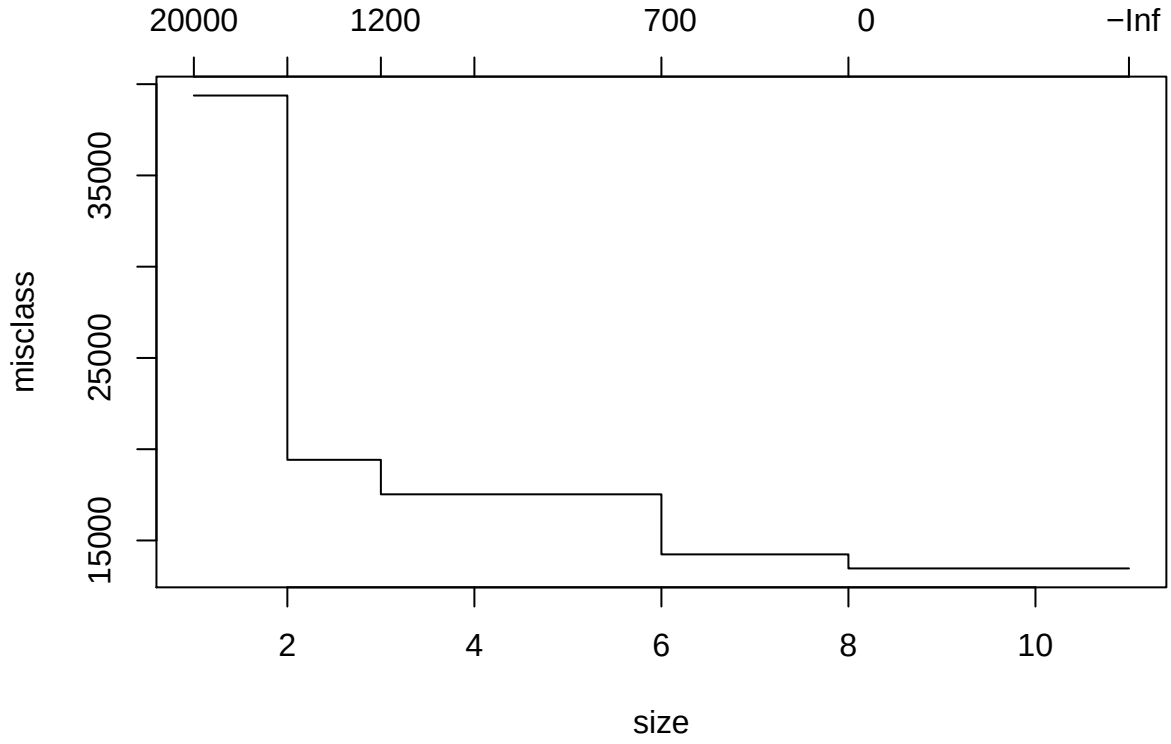
The first split is made according to the *online_boarding* using a threshold quite high, recalling the scale between 0 and 5 with which the variable is defined. The first two leaves are developed according two different variables and, in general,the split is often made on the basis of the *inflight_wifi_service*. Anyway,

in the summary are reported all the variables used, the number of terminal nodes and the miscassification rate, which is 0.1415 and, therefore, the tree classifier has an accuracy equal to 0.8585. This latter is shrunk with respect to the logistic regression model, but there are gains in the interpretation and understanding of the variables. Worth of attention is the role of *inflight_wifi_service*, whose behaviour seems puzzling: it classifies *satisfied* customers for values lower than 0.5 and making another split for values greater. This could highlight segment of customers which are not interested in the service and evaluate it with a lower grade because their satisfaction is driven by other factors, while lower-intermediate grade (like 1 or 2) could be evaluations from customers which are interested in an accurate rating of the service because of the main importance for their interest. After having looked at how variables are ranked, I decided to split the dataset into training and test sets to compute the prediction accuracy, which is almost equal to the one previously computed (0.8587), as the following table shows.

```
##                          airlines_test_labels
## tree_airlines_pred        neutral or dissatisfied satisfied
##   neutral or dissatisfied                   18369      1894
##   satisfied                                  3592     14991
```

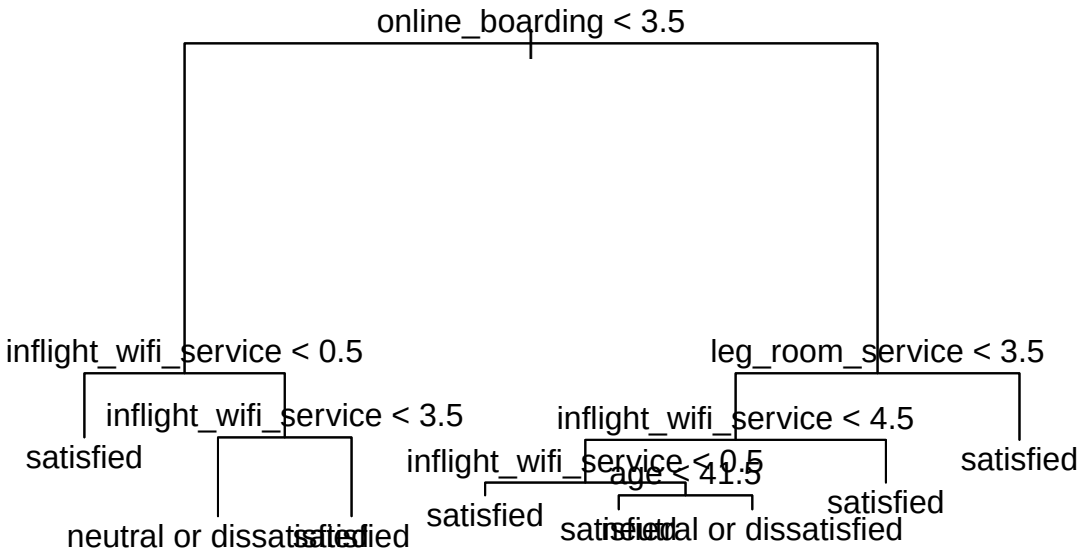
```
## [1] 85.87757
```

Since there are splits which do not modify the way in which the observations are classified, I tried to improve the accuracy performance of my tree predictor algorithm by deciding to prune it and by eliminating unuseful



splits.

The optimal number of terminal nodes selected by the pruning is 8, a point where the curve starts to begin flat until the maximum number of terminal nodes equal to 11. The pruned tree is reported below and its accuracy, computed with the code reported in the appendix, is equal to 0.8562, almost equal to the one of the complete

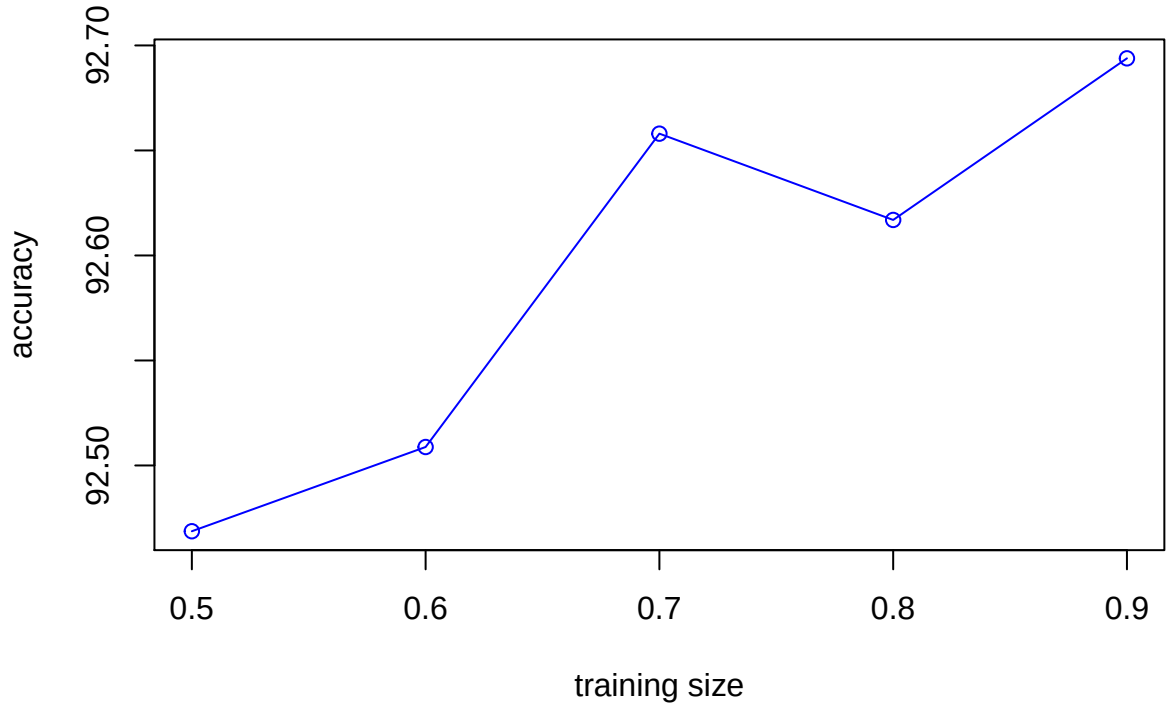


tree.

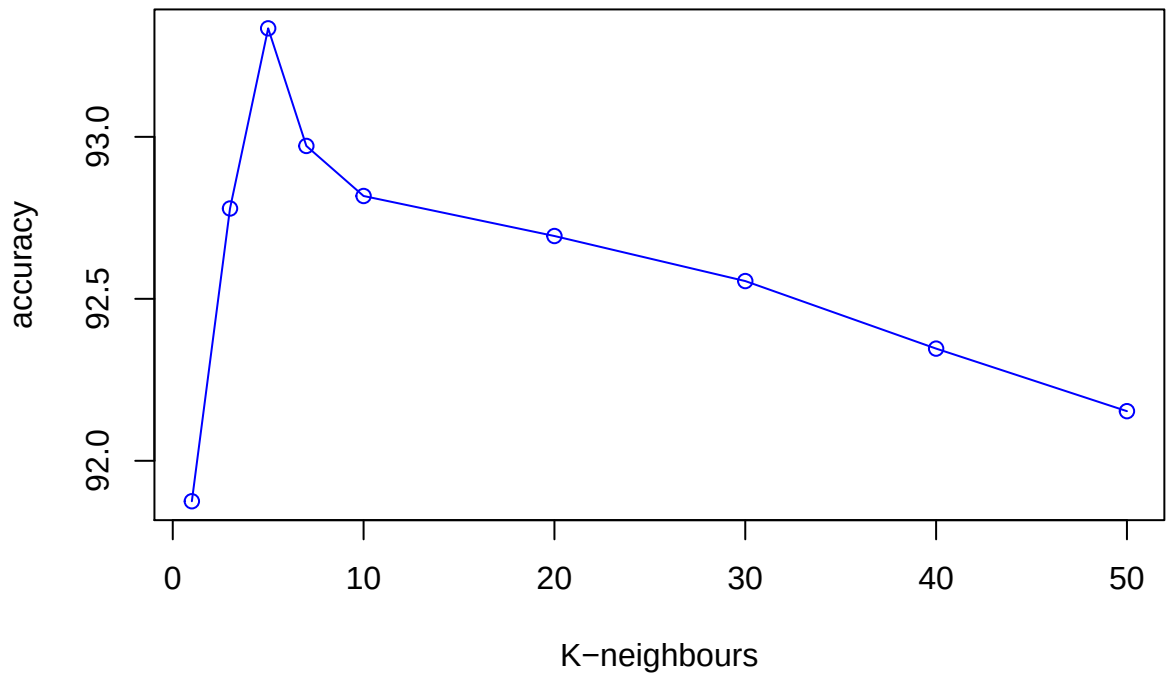
I tried also to run the same analysis starting with the usual dataset, but discarding the variables selected by the vif analysis of the previous section. At the end, the accuracy of the pruned tree is equal to 0.858, really near to the previous one. All the code to generate graphs and results is reported in the appendix. I tried to run the random forest using the tree classifier too, but, given the size of my dataset, this has been computationally impossible. For this reason, I used another algorithm to at least bring back the prediction accuracy to the levels of the logistic regression: **k-NN**.

K-NN

K-NN algorithm is here used for a classification problem and it assigns to each point the most common label among the k nearest points of the training set. The analysis presents good results, although computing training set accuracies and performing a cross-validation selection of k were infeasible due to computation reasons because the dataset is too large for the capacity of my tool. To run the algorithm, first I converted the character variables into integers and then I normalized them to weight the differences in the value ranges. Then, I started picking a random k (equal to 20) and a random split between training and test set sizes, with the training set size equal to 0.7 times respect to the starting dataset. The performance is good and better than trees: a test accuracy equal to 0.9265 is found. Then, I run a sort of learning curve analysis varying the training set size within the range including values *0.5, 0.6, 0.7, 0.8, 0.9* in order to find the best accuracy depending on the size of the training set. Even if slightly different, the results lay around the same small range, between 0.924 and 0.927, as shown in the following plot.



The performance with size equal to 0.9 seems to be slightly better, therefore I decided to move on considering this training size. The next step I performed is to analyze accuracy according to the variation of k in a sort of validation analysis. As before, I considered a finite set of possible values of k (1, 3, 5, 7, 10, 20, 30, 40, 50). The result is interesting, because it shows as the test accuracy is lower when k is equal to 1, probably because of too much variability and overfitting due to the difference with the training accuracy, which takes value 1. When k is equal to 5, the peak is reached and, going on with k values, first the accuracy jumps down until k is equal to 10 and then there is a constant decrease. Therefore, the best value of k among those I selected is $k = 5$, which, considering the training size equal to 0.9, takes a test accuracy value equal to 0.9333. This information is reported in the chart below.



I came back to the accuracy of the logistic regression, equal to 0.93. I tried to repeat the same step starting without considering the four selected variables which generate multicollinearity problem. The result is similar: the size is chosen equal to 0.8 with an accuracy equal to 0.9264, which becomes 0.9289 after having considered variations of k, chosing this latter equal to 7. The code is reported in the appendix.

Conclusions

Customer's loyalty and satisfaction are key features in the market airlines competition. This paper aims to study the relation between customer's satisfaction and several drivers considered relevant. Using logistic regression, tree predictors and k-NN algorithm, I tried to fit the data as well as possible. In the logistic regression, the model selected relevant variables useful to explain how and how much the probability to be satisfied is affected. Using a validation set approach, the accuracies of the training and the test sets are really great, respectively by taking values equal to 0.9344 and 0.9333. No overfitting is present, neither presence of underfitting. I add robustness to this first results in several ways: changing two times the random data partition and the training size, using a k-cross validation and by normalizing continuous variables present in the dataset, but the results stay close to the initial findings. After that, I have computed a tree classifier to improve the interpretation of my data and with the hope to have the same accuracy. Anyway, the results about this latter topic are not comfortable: equal to 0.8587, the misclassification rate is bigger than before because this tree algorithm with its splits in blocks does not fit well the data distribution. Anyway, looking at the tree, it is noteworthy the importance of the *online_boarding* as variable for the first split and then the widespread presence of the *inflight_wifi_service*. To improve the accuracy and reduce the variability, I pruned the tree, but the result of the test accuracy is not better: 0.85. In the end, I wanted to implement the random forest, but it has become not possible because of computational reasons for my tools, since the starting dataset contains almost 130000 observations. In order to obtain an accuracy at least equal to the one of the logistic regression, I used a k-NN algorithm. First, I normalized the data, then, starting

with the random value $k = 20$, I made variations to the training set size to find the best test accuracy. It came out that the training size should be equal to 0.9 respect to the starting dataset for an accuracy of 0.9260. Considering the value 0.9, I moved the number of k-neighbours to improve the test accuracy: the best performance is given by $k = 5$ with a test accuracy equal to 0.93334, as in the logistic regression. I could not compute training set accuracy and perform a k-cross validation always because of computation reasons given the dataset's size. In conclusion, these three algorithms give good predictive performances and select variables considered relevant from the customers' evaluation, identifying drivers of satisfaction.

Theoretical background

Logistic regression

The logistic regression is used to compute the probability that an event happens and it takes a form where the Euler constant is present and the result is always included between 0 and 1, no matter which and how many regressors are considered. To estimate the parameters *beta0, .. beta(n)*, the *maximum likelihood* is used to maximize the likelihood of the observed data. It is often used the logarithmic form to ease the computation, the so-called *logit*. In a problem wit two classes as here, the difference between the logistic regression and the discriminant analysis is that a discriminative learning is used in the first one, based on the conditional maximum likelihood ($Pr(Y|X)$), while LDA uses a generative learning with the full likelihood ($Pr(Y,X)$). Their results are often very similar, but LDA is not reported in the analysis because the number of observations is too big to perform a qualitative performance.

Trees

Used for both regression and classification problems, *tree-based methods* are so called because the set of splitting rules with which the predictor space is divided takes the form of a tree. The prediction space is divided in simple regions and the method is characterized by its interpretability, but it is usually less powerful than other approaches with respect to the accuracy. The tree is constructed with a *top-down* and *greedy* approach, respectively because it begins at the top of the tree with a unique set and then splitting the predictor space, and because at each step the split that adds the best possible improvement is made, without a strategy that would be careful to the future steps. The tree starts from the root and, passes through *internal nodes*, which represent the results of a splitting internal process, and arrives to its *final nodes*, also called *leaves*, which represent the resulting split of the feature space. The final space's division is composed by rectangles at the end for the sake of simplicity and interpretability, but the split could have any shape. This is part of the trade-off between accuracy and interpretability. The aim of the split is to minimize the variance within a group and, therefore, to find boxes which minimize the misclassification rate (or Gini's index or cross-entropy's index). A tree without limitation in its growing could guarantee good performances on the training set, but it might probably end up with overfitting. One possible solution is *to prune* it. The *pruning method* starts with a very large tree and then prune it back in order to obtain a subtree which perform better. This is done with the *cost complexity pruning*, which takes into consideration a parameter *alpha* such that, for each *alpha* value, a subtree is corresponded. *Alpha* controls a trade-off between the subtree's complexity and its fit to the training data and its optimal value is chosen using cross-validation. This process can face the overfitting problem, reducing the variance by cutting off not relevant features. In conclusion, trees and, in particular, pruned trees seem to reflect in the best realistic way the human decision-making process, they can be dispalyed graphically and allow for an easy interpreation.

K-nearest neighbours

The *k-NN nearest neighbours* algorithm is exploited in the analysis for a classification problem. This algorithm respond to a simple rule: it predicts every point with the label that is more present among the k neighbours of the point we are analyzing and, in case of tie, it responds to a predefined decision rule. The training size

plays its role in the prediction, therefore it is recommended to perform a learning curve analysis when a validation set approach is used. It is also relevant to focus on how it differently works depending on k, which takes value k = *1,..,* n with n equal to the size of the training set:

- *1-NN nearest neighbours* holds the best accuracy for point belonging to the training set, since each point would be predicted with its own label. Nevertheless, it provides bad performances on the test points with bad accuracy. For this reason, the case $k = 1$ is characterized by *overfitting* because of the difference between training and test errors and this is due to the too big variance.
- *K-NN* generalizes the previous particular case. The parameter k is usually chosen odd to avoid tie situations and if it is different from 1, in general the training error is different from 0. Moreover, as k grows, the classifiers generated become simpler. In particular, when $k = n$ the classifier is constant and equal to the most common label in the training set. Furthermore, as k increases too much, the algorithm ends up with an underfitting situation. In conclusion, there isn't a best value of k a priori, but it is usually set a level where the training error is different from 0.

K-NN works with binary classification problems and with multiclass classification problems using same modalities, but it works also in the regression problems, simply considering the prediction as the average of the labels of the k closest training points.

Appendix

```
library(plyr)
library(caret)
library(gridExtra)
library(tidyverse)
library(rsample)
library(e1071)
library(GGally)
library(data.table)
library(DT)
library(readr)
library(ggplot2)
library(dplyr)
library(tidyr)
library(corrplot)
library(rms)
library(MASS)
library(e1071)
library(ROCR)
library(gplots)
library(pROC)
library(ggpubr)

airlines <- read.csv("C:\\Users\\Utente\\OneDrive\\Desktop\\SL exam\\airline_passenger_satisfaction.csv
airlines['X'] <- NULL

# visualize dataset
str(airlines)
head(airlines)
```

```r
# Data preparation
airlines_clean <- airlines[complete.cases(airlines), ]
airlines_clean$satisfaction<-as.factor(airlines_clean$satisfaction)

##### primo tentativo (no factor)
# Data description
t=ggplot(airlines_clean, aes(x = satisfaction)) +
  geom_bar(aes(fill = satisfaction)) +
  geom_text(aes(y = ..count.. -200,
                label = paste0(round(prop.table(..count..),4) * 100, '%')),
            stat = 'count',
            position = position_dodge(.1),
            size = 3)
t
```

```r
#Customer categorical features and travel features
# Gender
t1 <- ggplot(airlines_clean, aes(x = Gender)) +
  geom_bar(aes(fill = satisfaction)) +
  geom_text(aes(y = ..count.. -200,
                label = paste0(round(prop.table(..count..),4) * 100, '%')),
            stat = 'count',
            position = position_dodge(.1),
            size = 3)

#Customer type
t2 <- ggplot(airlines_clean, aes(x = customer_type)) +
  geom_bar(aes(fill = satisfaction)) +
  geom_text(aes(y = ..count.. -200,
                label = paste0(round(prop.table(..count..),4) * 100, '%')),
            stat = 'count',
            position = position_dodge(.1),
            size = 3)

#Type of travel
t3 <- ggplot(airlines_clean, aes(x = type_of_travel)) +
  geom_bar(aes(fill = satisfaction)) +
  geom_text(aes(y = ..count.. -200,
                label = paste0(round(prop.table(..count..),4) * 100, '%')),
            stat = 'count',
            position = position_dodge(.1),
            size = 3)

#Customer class
t4 <- ggplot(airlines_clean, aes(x = customer_class)) +
  geom_bar(aes(fill = satisfaction)) +
  geom_text(aes(y = ..count.. -200,
                label = paste0(round(prop.table(..count..),4) * 100, '%')),
            stat = 'count',
            position = position_dodge(.1),
            size = 3)
```

```r
#Plot data within a grid
grid.arrange(t1, t2, t3, t4, ncol=1)
```

```r
# categorical variables evaluated by customers between 0 and 5 about fly experience

p1 <- ggplot(airlines_clean, aes(x = inflight_wifi_service)) +
  geom_bar(aes(fill = satisfaction)) +
  geom_text(aes(y = ..count.. -200,
                label = paste0(round(prop.table(..count..),4) * 100, '%')),
            stat = 'count',
            position = position_dodge(.1),
            size = 3)

p2 <- ggplot(airlines_clean, aes(x = departure_arrival_time_convenient)) +
  geom_bar(aes(fill = satisfaction)) +
  geom_text(aes(y = ..count.. -200,
                label = paste0(round(prop.table(..count..),4) * 100, '%')),
            stat = 'count',
            position = position_dodge(.1),
            size = 3)

p3 <- ggplot(airlines_clean, aes(x = ease_of_online_booking)) +
  geom_bar(aes(fill = satisfaction)) +
  geom_text(aes(y = ..count.. -200,
                label = paste0(round(prop.table(..count..),4) * 100, '%')),
            stat = 'count',
            position = position_dodge(.1),
            size = 3)

p4 <- ggplot(airlines_clean, aes(x = gate_location)) +
  geom_bar(aes(fill = satisfaction)) +
  geom_text(aes(y = ..count.. -200,
                label = paste0(round(prop.table(..count..),4) * 100, '%')),
            stat = 'count',
            position = position_dodge(.1),
            size = 3)

p5 <- ggplot(airlines_clean, aes(x = food_and_drink)) +
  geom_bar(aes(fill = satisfaction)) +
  geom_text(aes(y = ..count.. -200,
                label = paste0(round(prop.table(..count..),4) * 100, '%')),
            stat = 'count',
            position = position_dodge(.1),
            size = 3)

p6 <- ggplot(airlines_clean, aes(x = online_boarding)) +
  geom_bar(aes(fill = satisfaction)) +
  geom_text(aes(y = ..count.. -200,
                label = paste0(round(prop.table(..count..),4) * 100, '%')),
            stat = 'count',
            position = position_dodge(.1),
            size = 3)
```

```
p7 <- ggplot(airlines_clean, aes(x = seat_comfort)) +
  geom_bar(aes(fill = satisfaction)) +
  geom_text(aes(y = ..count.. -200,
                label = paste0(round(prop.table(..count..),4) * 100, '%')),
            stat = 'count',
            position = position_dodge(.1),
            size = 3)
p8 <- ggplot(airlines_clean, aes(x = inflight_entertainment)) +
  geom_bar(aes(fill = satisfaction)) +
  geom_text(aes(y = ..count.. -200,
                label = paste0(round(prop.table(..count..),4) * 100, '%')),
            stat = 'count',
            position = position_dodge(.1),
            size = 3)

p9 <- ggplot(airlines_clean, aes(x = onboard_service)) +
  geom_bar(aes(fill = satisfaction)) +
  geom_text(aes(y = ..count.. -200,
                label = paste0(round(prop.table(..count..),4) * 100, '%')),
            stat = 'count',
            position = position_dodge(.1),
            size = 3)

p10 <- ggplot(airlines_clean, aes(x = leg_room_service)) +
  geom_bar(aes(fill = satisfaction)) +
  geom_text(aes(y = ..count.. -200,
                label = paste0(round(prop.table(..count..),4) * 100, '%')),
            stat = 'count',
            position = position_dodge(.1),
            size = 3)

p11 <- ggplot(airlines_clean, aes(x = baggage_handling)) +
  geom_bar(aes(fill = satisfaction)) +
  geom_text(aes(y = ..count.. -200,
                label = paste0(round(prop.table(..count..),4) * 100, '%')),
            stat = 'count',
            position = position_dodge(.1),
            size = 3)

p12 <- ggplot(airlines_clean, aes(x = checkin_service)) +
  geom_bar(aes(fill = satisfaction)) +
  geom_text(aes(y = ..count.. -200,
                label = paste0(round(prop.table(..count..),4) * 100, '%')),
            stat = 'count',
            position = position_dodge(.1),
            size = 3)

p13 <- ggplot(airlines_clean, aes(x = inflight_service)) +
  geom_bar(aes(fill = satisfaction)) +
  geom_text(aes(y = ..count.. -200,
                label = paste0(round(prop.table(..count..),4) * 100, '%')),
            stat = 'count',
            position = position_dodge(.1),
```

```
            size = 3)

p14 <- ggplot(airlines_clean, aes(x = cleanliness)) +
  geom_bar(aes(fill = satisfaction)) +
  geom_text(aes(y = ..count.. -200,
                label = paste0(round(prop.table(..count..),4) * 100, '%')),
            stat = 'count',
            position = position_dodge(.1),
            size = 3)

grid.arrange(p1,p2,p3,p4,p5,p6,ncol=2)
```

```
grid.arrange(p7,p8,p9,p10,p11,p12,p13,p14, ncol=2)
```

```
# numerical variables
n1 <- ggplot(data = airlines_clean, aes(age, color = satisfaction))+
  geom_freqpoly(binwidth = 5, size = 1)

#Monthly charges histogram
n2 <- ggplot(data = airlines_clean, aes(flight_distance, color = satisfaction))+
  geom_freqpoly(binwidth = 5, size = 1)

#Total charges histogram
n3 <- ggplot(data = airlines_clean, aes(departure_delay_in_minutes, color = satisfaction))+
  geom_freqpoly(binwidth = 200, size = 1)

n4 <- ggplot(data = airlines_clean, aes(arrival_delay_in_minutes, color = satisfaction))+
  geom_freqpoly(binwidth = 200, size = 1)

#Plot quantitative data within a grid
grid.arrange(n1, n2, n3, n4,ncol=1)
```

```
######################### Logistic regression
# Making factors
airlines_clean$inflight_wifi_service <- as.character(airlines_clean$inflight_wifi_service)
airlines_clean$departure_arrival_time_convenient <- as.character(airlines_clean$departure_arrival_time_
airlines_clean$ease_of_online_booking <- as.character(airlines_clean$ease_of_online_booking)
airlines_clean$gate_location <- as.character(airlines_clean$gate_location)
airlines_clean$food_and_drink <- as.character(airlines_clean$food_and_drink)
airlines_clean$online_boarding <- as.character(airlines_clean$online_boarding)
airlines_clean$seat_comfort <- as.character(airlines_clean$seat_comfort)
airlines_clean$inflight_entertainment <- as.character(airlines_clean$inflight_entertainment)
airlines_clean$onboard_service <- as.character(airlines_clean$onboard_service)
airlines_clean$leg_room_service <- as.character(airlines_clean$leg_room_service)
airlines_clean$baggage_handling <- as.character(airlines_clean$baggage_handling)
airlines_clean$checkin_service <- as.character(airlines_clean$checkin_service)
airlines_clean$inflight_service <- as.character(airlines_clean$inflight_service)
airlines_clean$cleanliness <- as.character(airlines_clean$cleanliness)

airlines_clean <- subset(airlines_clean, airlines_clean$seat_comfort != 0)
airlines_clean <- subset(airlines_clean, airlines_clean$gate_location != 0)
```

```r
# Run regression
lr_fit <- glm(satisfaction~. , data = airlines_clean,
              family=binomial(link='logit'))
summary(lr_fit)

########## Detecting multicollinearity
library(performance)
check_collinearity(lr_fit)

# + ease_of_online_booking
lr_fit <- glm(satisfaction~.-ease_of_online_booking , airlines_clean,
              family=binomial(link='logit'))
lr_fit
check_collinearity(lr_fit)




# + inflight_entertainment
lr_fit <- glm(satisfaction~.-ease_of_online_booking-inflight_entertainment , data = airlines_clean,
              family=binomial(link='logit'))

lr_fit
check_collinearity(lr_fit)

# + arrival delay
lr_fit <- glm(satisfaction~.-ease_of_online_booking-inflight_entertainment-arrival_delay_in_minutes , da
              family=binomial(link='logit'))
lr_fit
check_collinearity(lr_fit)

# + departure_arrival_time_convenient
lr_fit <- glm(satisfaction~.-ease_of_online_booking-inflight_entertainment-arrival_delay_in_minutes-depa
              family=binomial(link='logit'))
summary(lr_fit)
check_collinearity(lr_fit)


#########
library(leaps)
# forward selection approach
regfit.fwd=regsubsets(satisfaction~.-ease_of_online_booking-inflight_entertainment-departure_delay_in_mi
reg.summary = summary(regfit.fwd)
plot(reg.summary$cp,xlab="Number of Variables",ylab="Cp")
```

```r
which.min(reg.summary$cp)

############ Predictions
set.seed(123)
split_train_test <- createDataPartition(airlines_clean$satisfaction,p=0.7,list=FALSE)
dtrain<- airlines_clean[split_train_test,]
dtest<-  airlines_clean[-split_train_test,]
lr_fit <- glm( satisfaction~.-ease_of_online_booking-inflight_entertainment-departure_delay_in_minutes-
```

```
              family=binomial(link='logit'))
summary(lr_fit)

# Confusion matrix


lr_prob1 <- predict(lr_fit, dtest, type="response")
lr_pred1 <- ifelse(lr_prob1 > 0.5,"Yes","No")
table(Predicted = lr_pred1, Actual = dtest$satisfaction)


# Accuracy
lr_prob2 <- predict(lr_fit, dtrain, type="response")
lr_pred2 <- ifelse(lr_prob2 > 0.5,"Yes","No")
lr_tab1 <- table(Predicted = lr_pred2, Actual = dtrain$satisfaction)
lr_tab2 <- table(Predicted = lr_pred1, Actual = dtest$satisfaction)

dtrain$satisfaction <- as.factor(mapvalues(dtrain$satisfaction,
                                           from = c('satisfied',
                                                    'neutral or dissatisfied'),
                                           to = c('Yes',
                                                  'No')))

dtest$satisfaction <- as.factor(mapvalues(dtest$satisfaction,
                                          from = c('satisfied',
                                                   'neutral or dissatisfied'),
                                          to = c('Yes',
                                                 'No')))
# Train
confusionMatrix(
  as.factor(lr_pred2),
  dtrain$satisfaction,
  positive = "Yes"
)
# Test
confusionMatrix(
  as.factor(lr_pred1),
  dtest$satisfaction,
  positive = "Yes"
)




######## Robustness changing data partition
set.seed(456)
split_train_test_rob <- createDataPartition(airlines_clean$satisfaction,p=0.7,list=FALSE)
dtrain_rob<- airlines_clean[split_train_test_rob,]
dtest_rob<-  airlines_clean[-split_train_test_rob,]
lr_fit_rob <- glm( satisfaction~.-ease_of_online_booking-inflight_entertainment-departure_delay_in_minu
                   family=binomial(link='logit'))
summary(lr_fit_rob)
```

```r
# Confusion matrix


lr_prob1_rob <- predict(lr_fit_rob, dtest_rob, type="response")
lr_pred1_rob <- ifelse(lr_prob1_rob > 0.5,"Yes","No")
table(Predicted = lr_pred1_rob, Actual = dtest_rob$satisfaction)


# Accuracy
lr_prob2_rob <- predict(lr_fit_rob, dtrain_rob, type="response")
lr_pred2_rob <- ifelse(lr_prob2_rob > 0.5,"Yes","No")
lr_tab1_rob <- table(Predicted = lr_pred2_rob, Actual = dtrain_rob$satisfaction)
lr_tab2_rob <- table(Predicted = lr_pred1_rob, Actual = dtest_rob$satisfaction)

dtrain_rob$satisfaction <- as.factor(mapvalues(dtrain_rob$satisfaction,
                                               from = c('satisfied',
                                                        'neutral or dissatisfied'),
                                               to = c('Yes',
                                                      'No')))

dtest_rob$satisfaction <- as.factor(mapvalues(dtest_rob$satisfaction,
                                              from = c('satisfied',
                                                       'neutral or dissatisfied'),
                                              to = c('Yes',
                                                     'No')))
# Train
confusionMatrix(
  as.factor(lr_pred2_rob),
  dtrain_rob$satisfaction,
  positive = "Yes"
)
# Test
confusionMatrix(
  as.factor(lr_pred1_rob),
  dtest_rob$satisfaction,
  positive = "Yes"
)

########## Third different data partition
set.seed(789)
split_train_test_rob1 <- createDataPartition(airlines_clean$satisfaction,p=0.7,list=FALSE)
dtrain_rob1<- airlines_clean[split_train_test_rob1,]
dtest_rob1<-  airlines_clean[-split_train_test_rob1,]
lr_fit_rob1 <- glm(satisfaction~.-ease_of_online_booking-inflight_entertainment-departure_delay_in_minu
                   family=binomial(link='logit'))
summary(lr_fit_rob)

# Confusion matrix


lr_prob1_rob1 <- predict(lr_fit_rob1, dtest_rob1, type="response")
lr_pred1_rob1 <- ifelse(lr_prob1_rob1 > 0.5,"Yes","No")
table(Predicted = lr_pred1_rob1, Actual = dtest_rob1$satisfaction)
```

```r
# Accuracy
lr_prob2_rob1 <- predict(lr_fit_rob1, dtrain_rob1, type="response")
lr_pred2_rob1 <- ifelse(lr_prob2_rob1 > 0.5,"Yes","No")
lr_tab1_rob1 <- table(Predicted = lr_pred2_rob1, Actual = dtrain_rob1$satisfaction)
lr_tab2_rob1 <- table(Predicted = lr_pred1_rob1, Actual = dtest_rob1$satisfaction)

dtrain_rob1$satisfaction <- as.factor(mapvalues(dtrain_rob1$satisfaction,
                                                from = c('satisfied',
                                                         'neutral or dissatisfied'),
                                                to = c('Yes',
                                                       'No')))

dtest_rob1$satisfaction <- as.factor(mapvalues(dtest_rob1$satisfaction,
                                               from = c('satisfied',
                                                        'neutral or dissatisfied'),
                                               to = c('Yes',
                                                      'No')))
# Train
confusionMatrix(
  as.factor(lr_pred2_rob1),
  dtrain_rob1$satisfaction,
  positive = "Yes"
)
# Test
confusionMatrix(
  as.factor(lr_pred1_rob1),
  dtest_rob1$satisfaction,
  positive = "Yes"
)

### Training size equal to 0.9
set.seed(555)
split_train_test <- createDataPartition(airlines_clean$satisfaction,p=0.9,list=FALSE)
dtrain<- airlines_clean[split_train_test,]
dtest<-  airlines_clean[-split_train_test,]
lr_fit <- glm(satisfaction~.-ease_of_online_booking-inflight_entertainment-departure_delay_in_minutes-d
              family=binomial(link='logit'))
summary(lr_fit)

# Confusion matrix


lr_prob1 <- predict(lr_fit, dtest, type="response")
lr_pred1 <- ifelse(lr_prob1 > 0.5,"Yes","No")
table(Predicted = lr_pred1, Actual = dtest$satisfaction)


# Accuracy
lr_prob2 <- predict(lr_fit, dtrain, type="response")
lr_pred2 <- ifelse(lr_prob2 > 0.5,"Yes","No")
lr_tab1 <- table(Predicted = lr_pred2, Actual = dtrain$satisfaction)
lr_tab2 <- table(Predicted = lr_pred1, Actual = dtest$satisfaction)
```

```r
dtrain$satisfaction <- as.factor(mapvalues(dtrain$satisfaction,
                                           from = c('satisfied',
                                                    'neutral or dissatisfied'),
                                           to = c('Yes',
                                                  'No')))

dtest$satisfaction <- as.factor(mapvalues(dtest$satisfaction,
                                          from = c('satisfied',
                                                   'neutral or dissatisfied'),
                                          to = c('Yes',
                                                 'No')))
# Train
confusionMatrix(
  as.factor(lr_pred2),
  dtrain$satisfaction,
  positive = "Yes"
)
# Test
confusionMatrix(
  as.factor(lr_pred1),
  dtest$satisfaction,
  positive = "Yes"
)

######## k-cross-validation
# Define training control
set.seed(124)
train.control <- trainControl(method = "cv", number = 10)
# Train the model
model <- train( satisfaction~.-ease_of_online_booking-inflight_entertainment-departure_delay_in_minutes
               trControl = train.control)
# Summarize the results
print(model)


####### With normalized variables

# First split

set.seed(999)
split_train_test_norm <- createDataPartition(airlines_clean$satisfaction,p=0.7,list=FALSE)
dtrain_nor<- airlines_clean[split_train_test_norm,]
dtest_nor<-  airlines_clean[-split_train_test_norm,]

# normalize training set
dist_scaled <- (dtrain_nor$flight_distance-mean(dtrain_nor$flight_distance))/(max(dtrain_nor$flight_dis
age_scaled <- (dtrain_nor$age-mean(dtrain_nor$age))/(max(dtrain_nor$age) - min(dtrain_nor$age))
dep_scaled <- (dtrain_nor$departure_delay_in_minutes-mean(dtrain_nor$departure_delay_in_minutes))/(max(
arr_scaled <- (dtrain_nor$arrival_delay_in_minutes-mean(dtrain_nor$arrival_delay_in_minutes))/(max(dtra
dtrain_nor$flight_distance <- dist_scaled
dtrain_nor$age <- age_scaled
dtrain_nor$arrival_delay_in_minutes <- arr_scaled
```

```r
dtrain_norm<- dtrain_nor
# normalize test set
dist_scaled <- (dtest_nor$flight_distance-mean(dtest_nor$flight_distance))/(max(dtest_nor$flight_distanc
age_scaled <- (dtest_nor$age-mean(dtest_nor$age))/(max(dtest_nor$age) - min(dtest_nor$age))
dep_scaled <- (dtest_nor$departure_delay_in_minutes-mean(dtest_nor$departure_delay_in_minutes))/(max(dt
arr_scaled <- (dtest_nor$arrival_delay_in_minutes-mean(dtest_nor$arrival_delay_in_minutes))/(max(dtest_
dtest_nor$flight_distance <- dist_scaled
dtest_nor$age <- age_scaled
dtest_nor$arrival_delay_in_minutes <- arr_scaled

dtest_norm<-  dtest_nor

lr_fit_norm <- glm(satisfaction~.-ease_of_online_booking-inflight_entertainment-departure_delay_in_minu
                   family=binomial(link='logit'))


# Confusion matrix


lr_prob1_norm <- predict(lr_fit_norm, dtest_norm, type="response")
lr_pred1_norm <- ifelse(lr_prob1_norm > 0.5,"Yes","No")
table(Predicted = lr_pred1_norm, Actual = dtest_norm$satisfaction)


# Accuracy
lr_prob2_norm <- predict(lr_fit_norm, dtrain_norm, type="response")
lr_pred2_norm <- ifelse(lr_prob2_norm > 0.5,"Yes","No")
lr_tab1_norm <- table(Predicted = lr_pred2_norm, Actual = dtrain_norm$satisfaction)
lr_tab2_norm <- table(Predicted = lr_pred1_norm, Actual = dtest_norm$satisfaction)

dtrain_norm$satisfaction <- as.factor(mapvalues(dtrain_norm$satisfaction,
                                                from = c('satisfied',
                                                         'neutral or dissatisfied'),
                                                to = c('Yes',
                                                       'No')))

dtest_norm$satisfaction <- as.factor(mapvalues(dtest_norm$satisfaction,
                                               from = c('satisfied',
                                                        'neutral or dissatisfied'),
                                               to = c('Yes',
                                                      'No')))
# Train
confusionMatrix(
  as.factor(lr_pred2_norm),
  dtrain_norm$satisfaction,
  positive = "Yes"
)
# Test
confusionMatrix(
  as.factor(lr_pred1_norm),
  dtest_norm$satisfaction,
  positive = "Yes"
)
```

```r
############# Second split
set.seed(888)
split_train_test_norm <- createDataPartition(airlines_clean$satisfaction,p=0.7,list=FALSE)
dtrain_nor<- airlines_clean[split_train_test_norm,]
dtest_nor<-  airlines_clean[-split_train_test_norm,]
# normalize training set
dist_scaled <- (dtrain_nor$flight_distance-mean(dtrain_nor$flight_distance))/(max(dtrain_nor$flight_dis
age_scaled <- (dtrain_nor$age-mean(dtrain_nor$age))/(max(dtrain_nor$age) - min(dtrain_nor$age))
dep_scaled <- (dtrain_nor$departure_delay_in_minutes-mean(dtrain_nor$departure_delay_in_minutes))/(max(
arr_scaled <- (dtrain_nor$arrival_delay_in_minutes-mean(dtrain_nor$arrival_delay_in_minutes))/(max(dtra
dtrain_nor$flight_distance <- dist_scaled
dtrain_nor$age <- age_scaled
dtrain_nor$departure_delay_in_minutes <- dep_scaled
dtrain_nor$arrival_delay_in_minutes <- arr_scaled

dtrain_norm<- dtrain_nor
# normalize test set
dist_scaled <- (dtest_nor$flight_distance-mean(dtest_nor$flight_distance))/(max(dtest_nor$flight_distan
age_scaled <- (dtest_nor$age-mean(dtest_nor$age))/(max(dtest_nor$age) - min(dtest_nor$age))
dep_scaled <- (dtest_nor$departure_delay_in_minutes-mean(dtest_nor$departure_delay_in_minutes))/(max(dt
arr_scaled <- (dtest_nor$arrival_delay_in_minutes-mean(dtest_nor$arrival_delay_in_minutes))/(max(dtest_
dtest_nor$flight_distance <- dist_scaled
dtest_nor$age <- age_scaled
dtest_nor$departure_delay_in_minutes <- dep_scaled
dtest_nor$arrival_delay_in_minutes <- arr_scaled

dtest_norm<-  dtest_nor

lr_fit_norm <- glm(satisfaction~.-ease_of_online_booking-inflight_entertainment-departure_delay_in_minu
                   family=binomial(link='logit'))

summary(lr_fit_norm)

# Confusion matrix


lr_prob1_norm <- predict(lr_fit_norm, dtest_norm, type="response")
lr_pred1_norm <- ifelse(lr_prob1_norm > 0.5,"Yes","No")
table(Predicted = lr_pred1_norm, Actual = dtest_norm$satisfaction)


# Accuracy
lr_prob2_norm <- predict(lr_fit_norm, dtrain_norm, type="response")
lr_pred2_norm <- ifelse(lr_prob2_norm > 0.5,"Yes","No")
lr_tab1_norm <- table(Predicted = lr_pred2_norm, Actual = dtrain_norm$satisfaction)
lr_tab2_norm <- table(Predicted = lr_pred1_norm, Actual = dtest_norm$satisfaction)

dtrain_norm$satisfaction <- as.factor(mapvalues(dtrain_norm$satisfaction,
                                                from = c('satisfied',
                                                         'neutral or dissatisfied'),
                                                to = c('Yes',
                                                       'No')))
```

```r
dtest_norm$satisfaction <- as.factor(mapvalues(dtest_norm$satisfaction,
                                               from = c('satisfied',
                                                        'neutral or dissatisfied'),
                                               to = c('Yes',
                                                      'No')))
# Train
confusionMatrix(
  as.factor(lr_pred2_norm),
  dtrain_norm$satisfaction,
  positive = "Yes"
)
# Test
confusionMatrix(
  as.factor(lr_pred1_norm),
  dtest_norm$satisfaction,
  positive = "Yes"
)



####################################
# With cross-validation
dist_scaled <- (airlines_clean$flight_distance-mean(airlines_clean$flight_distance))/(max(airlines_clean
age_scaled <- (airlines_clean$age-mean(airlines_clean$age))/(max(airlines_clean$age) - min(airlines_clea
dep_scaled <- (airlines_clean$departure_delay_in_minutes-mean(airlines_clean$departure_delay_in_minutes)
arr_scaled <- (airlines_clean$arrival_delay_in_minutes-mean(airlines_clean$arrival_delay_in_minutes))/(m
airlines_clean$flight_distance <- dist_scaled
airlines_clean$age <- age_scaled
airlines_clean$departure_delay_in_minutes <- dep_scaled
airlines_clean$arrival_delay_in_minutes <- arr_scaled

set.seed(777)
train.control_norm <- trainControl(method = "cv", number = 10)
# Train the model
model_norm <- train(satisfaction~.-ease_of_online_booking-inflight_entertainment-departure_delay_in_minu
                    trControl = train.control_norm)
# Summarize the results
print(model_norm)


#############################################################
########## Recover the starting dataset

airlines <- read.csv("C:\\Users\\Utente\\OneDrive\\Desktop\\SL exam\\airline_passenger_satisfaction.csv")
airlines['X'] <- NULL

# Data preparation
airlines_clean <- airlines[complete.cases(airlines), ]
airlines_clean$satisfaction<-as.factor(airlines_clean$satisfaction)
airlines_clean <- subset(airlines_clean, airlines_clean$seat_comfort != 0)
########## Be careful of error

###############Tree
```

```r
library(tree)
library(ISLR)

tree.airlines <- tree(satisfaction~., data = airlines_clean)
summary(tree.airlines)
plot(tree.airlines)
text(tree.airlines, pretty = 0)
```

```r
tree.airlines

# Accuracy 0.85
####### Trying to split train-test
set.seed(101)
train = sample(1:nrow(airlines_clean), 0.7*nrow(airlines_clean))
airlines_train = airlines_clean[train,-23]
airlines_test = airlines_clean[-train,-23]
airlines_train_labels <- airlines_clean[train, 23]
airlines_test_labels <- airlines_clean[-train, 23]
summary(airlines_train_labels)
summary(airlines_test_labels)

tree_airlines_train <- tree(satisfaction~., airlines_clean[train,])
tree_airlines_pred <- predict(tree_airlines_train, airlines_clean[-train,], type = 'class')
table(tree_airlines_pred, airlines_test_labels)

accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
accuracy(table(tree_airlines_pred, airlines_test_labels))

# 0.85
######### Pruning it

set.seed(101)
cv.airlines=cv.tree(tree_airlines_train,FUN=prune.misclass)
cv.airlines
plot(cv.airlines) #8
```

```r
prune.airlines=prune.misclass(tree_airlines_train,best=8)
plot(prune.airlines);text(prune.airlines,pretty=0)
```

```r
tree.air.pred=predict(prune.airlines,airlines_clean[-train,],type="class")
with(airlines_clean[-train,],table(tree.air.pred,satisfaction))

### Accuracy 0.85

########### Repeat without discarded variables


tree.airlines <- tree(satisfaction~.-ease_of_online_booking-inflight_entertainment-departure_delay_in_m
summary(tree.airlines)
plot(tree.airlines)
text(tree.airlines, pretty = 0)
```

```
tree.airlines


####### Trying to split train-test
set.seed(202)
train = sample(1:nrow(airlines_clean), 0.7*nrow(airlines_clean))
airlines_train = airlines_clean[train,-23]
airlines_test = airlines_clean[-train,-23]
airlines_train_labels <- airlines_clean[train, 23]
airlines_test_labels <- airlines_clean[-train, 23]
summary(airlines_train_labels)
summary(airlines_test_labels)

tree_airlines_train <- tree(satisfaction~.-ease_of_online_booking-inflight_entertainment-departure_dela
tree_airlines_pred <- predict(tree_airlines_train, airlines_clean[-train,], type = 'class')
table(tree_airlines_pred, airlines_test_labels)

accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
accuracy(table(tree_airlines_pred, airlines_test_labels))


######### Pruning it

set.seed(202)
cv.airlines=cv.tree(tree_airlines_train,FUN=prune.misclass)
cv.airlines
plot(cv.airlines)
```

```
prune.airlines=prune.misclass(tree_airlines_train,best=8)
plot(prune.airlines);text(prune.airlines,pretty=0)
```

```
tree.air.pred=predict(prune.airlines,airlines_clean[-train,],type="class")
with(airlines_clean[-train,],table(tree.air.pred,satisfaction))



################################## K-NN
library(class)

airlines_clean$inflight_wifi_service <- as.integer(airlines_clean$inflight_wifi_service)
airlines_clean$departure_arrival_time_convenient <- as.integer(airlines_clean$departure_arrival_time_co
airlines_clean$ease_of_online_booking <- as.integer(airlines_clean$ease_of_online_booking)
airlines_clean$gate_location <- as.integer(airlines_clean$gate_location)
airlines_clean$food_and_drink <- as.integer(airlines_clean$food_and_drink)
airlines_clean$online_boarding <- as.integer(airlines_clean$online_boarding)
airlines_clean$seat_comfort <- as.integer(airlines_clean$seat_comfort)
airlines_clean$inflight_entertainment <- as.integer(airlines_clean$inflight_entertainment)
airlines_clean$onboard_service <- as.integer(airlines_clean$onboard_service)
airlines_clean$leg_room_service <- as.integer(airlines_clean$leg_room_service)
airlines_clean$baggage_handling <- as.integer(airlines_clean$baggage_handling)
airlines_clean$checkin_service <- as.integer(airlines_clean$checkin_service)
airlines_clean$inflight_service <- as.integer(airlines_clean$inflight_service)
airlines_clean$cleanliness <- as.integer(airlines_clean$cleanliness)
```

```r
airlines_clean$Gender <- as.integer(mapvalues(airlines_clean$Gender,
                                              from = c('Female',
                                                       'Male'),
                                              to = c('0' , '1')))
airlines_clean$customer_type <- as.integer(mapvalues(airlines_clean$customer_type,
                                                     from = c('disloyal Customer',
                                                              'Loyal Customer'),
                                                     to = c('0' , '1')))
airlines_clean$type_of_travel <- as.integer(mapvalues(airlines_clean$type_of_travel,
                                                      from = c('Personal Travel',
                                                               'Business travel'),
                                                      to = c('0' , '1')))
airlines_clean$customer_class <- as.integer(mapvalues(airlines_clean$customer_class,
                                                      from = c('Eco',
                                                               'Eco Plus',
                                                               'Business'),
                                                      to = c('0' , '1', '2')))
airlines_clean$satisfaction <- as.integer(mapvalues(airlines_clean$satisfaction,
                                                    from = c('neutral or dissatisfied',
                                                             'satisfied'),
                                                    to = c('0' , '1')))




normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}

accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}

dt <- as.data.frame(lapply(airlines_clean, normalize))

######## size 0.7
set.seed(12)
train.index7 <- createDataPartition(dt$satisfaction, p = .7, list = FALSE)
dt_train7 <- dt[ train.index7,-23]
dt_test7  <- dt[-train.index7,-23]

dt_train_labels7 <- dt[train.index7, 23]
dt_test_labels7 <- dt[-train.index7, 23]

table17 <- table(dt_train_labels7)
round(prop.table(table17), 2)

table27 <- table(dt_test_labels7)
round(prop.table(table27), 2)


##run knn function
pr7 <- knn(dt_train7,dt_test7,cl=dt_train_labels7,k=20)

##create confusion matrix
```

```
tab7 <- table(pr7,dt_test_labels7)

#### size 0.6
set.seed(34)
train.index6 <- createDataPartition(dt$satisfaction, p = .6, list = FALSE)
dt_train6 <- dt[ train.index6,-23]
dt_test6  <- dt[-train.index6,-23]

dt_train_labels6 <- dt[train.index6, 23]
dt_test_labels6 <- dt[-train.index6, 23]

table16 <- table(dt_train_labels6)
round(prop.table(table16), 2)

table26 <- table(dt_test_labels6)
round(prop.table(table26), 2)


##run knn function
pr6 <- knn(dt_train6,dt_test6,cl=dt_train_labels6,k=20)

##create confusion matrix
tab6 <- table(pr6,dt_test_labels6)



#### size 0.8
set.seed(56)
train.index8 <- createDataPartition(dt$satisfaction, p = .8, list = FALSE)
dt_train8 <- dt[ train.index8,-23]
dt_test8  <- dt[-train.index8,-23]

dt_train_labels8 <- dt[train.index8, 23]
dt_test_labels8 <- dt[-train.index8, 23]

table18 <- table(dt_train_labels8)
round(prop.table(table18), 2)

table28 <- table(dt_test_labels8)
round(prop.table(table28), 2)


##run knn function
pr8 <- knn(dt_train8,dt_test8,cl=dt_train_labels8,k=20)

##create confusion matrix
tab8 <- table(pr8,dt_test_labels8)

##this function divides the correct predictions by total number of predictions that tell us how accurat

accuracy(tab8)

# size 0.9
```

```r
set.seed(78)
train.index9 <- createDataPartition(dt$satisfaction, p = .9, list = FALSE)
dt_train9 <- dt[ train.index9,-23]
dt_test9  <- dt[-train.index9,-23]

dt_train_labels9 <- dt[train.index9, 23]
dt_test_labels9 <- dt[-train.index9, 23]

table19 <- table(dt_train_labels9)
round(prop.table(table19), 2)

table29 <- table(dt_test_labels9)
round(prop.table(table29), 2)


##run knn function
pr9 <- knn(dt_train9,dt_test9,cl=dt_train_labels9,k=20)

##create confusion matrix
tab9 <- table(pr9,dt_test_labels9)

##this function divides the correct predictions by total number of predictions that tell us how accurat


# size 0.5
set.seed(90)
train.index5 <- createDataPartition(dt$satisfaction, p = .5, list = FALSE)
dt_train5 <- dt[ train.index5,-23]
dt_test5  <- dt[-train.index5,-23]

dt_train_labels5 <- dt[train.index5, 23]
dt_test_labels5 <- dt[-train.index5, 23]

table15 <- table(dt_train_labels5)
round(prop.table(table15), 2)

table25 <- table(dt_test_labels5)
round(prop.table(table25), 2)


##run knn function
pr5 <- knn(dt_train5,dt_test5,cl=dt_train_labels5,k=20)

##create confusion matrix
tab5 <- table(pr5,dt_test_labels5)

##this function divides the correct predictions by total number of predictions that tell us how accurat




################ plot based on training size
accuracies <- c(accuracy(tab5), accuracy(tab6), accuracy(tab7),accuracy(tab8), accuracy(tab9) )
```

```r
thicks <- c(0.5,0.6,0.7,0.8,0.9)

plot(thicks, accuracies, type="o", col="blue", xlab = "training size", ylab = "accuracy")
```

```r
################## Accuracies are pretty stable around 92.60. The best one is that with training size 0


#K = 1
prK1 <- knn(dt_train9,dt_test9,cl=dt_train_labels9,k=1)
##create confusion matrix
tabK1 <- table(prK1,dt_test_labels9)

#K = 3
prK3 <- knn(dt_train9,dt_test9,cl=dt_train_labels9,k=3)
##create confusion matrix
tabK3 <- table(prK3,dt_test_labels9)

#K = 5
prK5 <- knn(dt_train9,dt_test9,cl=dt_train_labels9,k=5)
##create confusion matrix
tabK5 <- table(prK5,dt_test_labels9)

# K = 7
prK7 <- knn(dt_train9,dt_test9,cl=dt_train_labels9,k=7)
##create confusion matrix
tabK7 <- table(prK7,dt_test_labels9)

## k = 10
pr10 <- knn(dt_train9,dt_test9,cl=dt_train_labels9,k=10)
##create confusion matrix
tabK10 <- table(pr10,dt_test_labels9)



## K = 30
pr30 <- knn(dt_train9,dt_test9,cl=dt_train_labels9,k=30)
##create confusion matrix
tab30 <- table(pr30,dt_test_labels9)


## K = 40
pr40 <- knn(dt_train9,dt_test9,cl=dt_train_labels9,k=40)
##create confusion matrix
tab40 <- table(pr40,dt_test_labels9)


## K = 50
pr50 <- knn(dt_train9,dt_test9,cl=dt_train_labels9,k=50)
##create confusion matrix
tab50 <- table(pr50,dt_test_labels9)


accuraciesK <- c(accuracy(tabK1), accuracy(tabK3), accuracy(tabK5),accuracy(tabK7), accuracy(tab10), ac
```

```r
thicksK <- c(1,3,5,7,10,20,30,40,50)

plot(thicksK, accuraciesK, type="o", col="blue", xlab = "K-neighbours", ylab = "accuracy")
```

```r
###################### K = 5 best neighbours with accuracy93.334 and tr. size = 0.9


############ repeat the analysis without the discarded variables because of multicollinearity

dt <- dt[,-c(8,9,14,21)]

######## size 0.7
set.seed(1212)
train.index7 <- createDataPartition(dt$satisfaction, p = .7, list = FALSE)
dt_train7 <- dt[ train.index7,-19]
dt_test7  <- dt[-train.index7,-19]

dt_train_labels7 <- dt[train.index7, 19]
dt_test_labels7 <- dt[-train.index7, 19]

table17 <- table(dt_train_labels7)
round(prop.table(table17), 2)

table27 <- table(dt_test_labels7)
round(prop.table(table27), 2)


##run knn function
pr7 <- knn(dt_train7,dt_test7,cl=dt_train_labels7,k=20)

##create confusion matrix
tab7 <- table(pr7,dt_test_labels7)

#### size 0.6
set.seed(3434)
train.index6 <- createDataPartition(dt$satisfaction, p = .6, list = FALSE)
dt_train6 <- dt[ train.index6,-19]
dt_test6  <- dt[-train.index6,-19]

dt_train_labels6 <- dt[train.index6, 19]
dt_test_labels6 <- dt[-train.index6, 19]

table16 <- table(dt_train_labels6)
round(prop.table(table16), 2)

table26 <- table(dt_test_labels6)
round(prop.table(table26), 2)


##run knn function
pr6 <- knn(dt_train6,dt_test6,cl=dt_train_labels6,k=20)

##create confusion matrix
```

```
tab6 <- table(pr6,dt_test_labels6)



#### size 0.8
set.seed(5656)
train.index8 <- createDataPartition(dt$satisfaction, p = .8, list = FALSE)
dt_train8 <- dt[ train.index8,-19]
dt_test8  <- dt[-train.index8,-19]

dt_train_labels8 <- dt[train.index8, 19]
dt_test_labels8 <- dt[-train.index8, 19]

table18 <- table(dt_train_labels8)
round(prop.table(table18), 2)

table28 <- table(dt_test_labels8)
round(prop.table(table28), 2)


##run knn function
pr8 <- knn(dt_train8,dt_test8,cl=dt_train_labels8,k=20)

##create confusion matrix
tab8 <- table(pr8,dt_test_labels8)

##this function divides the correct predictions by total number of predictions that tell us how accurat

accuracy(tab8)

# size 0.9
set.seed(7878)
train.index9 <- createDataPartition(dt$satisfaction, p = .9, list = FALSE)
dt_train9 <- dt[ train.index9,-19]
dt_test9  <- dt[-train.index9,-19]

dt_train_labels9 <- dt[train.index9, 19]
dt_test_labels9 <- dt[-train.index9, 19]

table19 <- table(dt_train_labels9)
round(prop.table(table19), 2)

table29 <- table(dt_test_labels9)
round(prop.table(table29), 2)


##run knn function
pr9 <- knn(dt_train9,dt_test9,cl=dt_train_labels9,k=20)

##create confusion matrix
tab9 <- table(pr9,dt_test_labels9)

##this function divides the correct predictions by total number of predictions that tell us how accurat
```

```r
# size 0.5
set.seed(9090)
train.index5 <- createDataPartition(dt$satisfaction, p = .5, list = FALSE)
dt_train5 <- dt[ train.index5,-19]
dt_test5  <- dt[-train.index5,-19]

dt_train_labels5 <- dt[train.index5, 19]
dt_test_labels5 <- dt[-train.index5, 19]

table15 <- table(dt_train_labels5)
round(prop.table(table15), 2)

table25 <- table(dt_test_labels5)
round(prop.table(table25), 2)


##run knn function
pr5 <- knn(dt_train5,dt_test5,cl=dt_train_labels5,k=20)

##create confusion matrix
tab5 <- table(pr5,dt_test_labels5)

##this function divides the correct predictions by total number of predictions that tell us how accurat




################ plot based on training size
accuracies <- c(accuracy(tab5), accuracy(tab6), accuracy(tab7),accuracy(tab8), accuracy(tab9) )
thicks <- c(0.5,0.6,0.7,0.8,0.9)

plot(thicks, accuracies, type="o", col="blue", xlab = "training size", ylab = "accuracy")
```

```r
################## Accuracies are pretty stable around 92.60. The best one is that with training size 0


#K = 1
prK1 <- knn(dt_train8,dt_test8,cl=dt_train_labels8,k=1)
##create confusion matrix
tabK1 <- table(prK1,dt_test_labels8)

#K = 3
prK3 <- knn(dt_train8,dt_test8,cl=dt_train_labels8,k=3)
##create confusion matrix
tabK3 <- table(prK3,dt_test_labels8)

#K = 5
prK5 <- knn(dt_train8,dt_test8,cl=dt_train_labels8,k=5)
##create confusion matrix
tabK5 <- table(prK5,dt_test_labels8)

# K = 7
```

```
prK7 <- knn(dt_train8,dt_test8,cl=dt_train_labels8,k=7)
##create confusion matrix
tabK7 <- table(prK7,dt_test_labels8)

## k = 10
pr10 <- knn(dt_train8,dt_test8,cl=dt_train_labels8,k=10)
##create confusion matrix
tabK10 <- table(pr10,dt_test_labels8)



## K = 30
pr30 <- knn(dt_train8,dt_test8,cl=dt_train_labels8,k=30)
##create confusion matrix
tab30 <- table(pr30,dt_test_labels8)


## K = 40
pr40 <- knn(dt_train8,dt_test8,cl=dt_train_labels8,k=40)
##create confusion matrix
tab40 <- table(pr40,dt_test_labels8)


## K = 50
pr50 <- knn(dt_train8,dt_test8,cl=dt_train_labels8,k=50)
##create confusion matrix
tab50 <- table(pr50,dt_test_labels8)


accuraciesK <- c(accuracy(tabK1), accuracy(tabK3), accuracy(tabK5),accuracy(tabK7), accuracy(tabK10), a
thicksK <- c(1,3,5,7,10,20,30,40,50)

plot(thicksK, accuraciesK, type="o", col="blue", xlab = "K-neighbours", ylab = "accuracy")
```