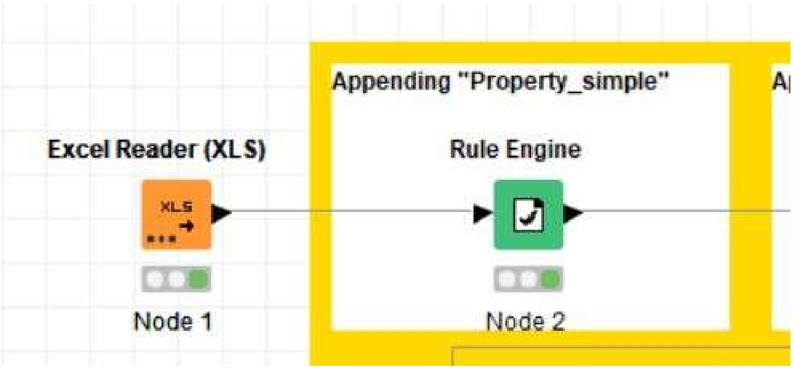
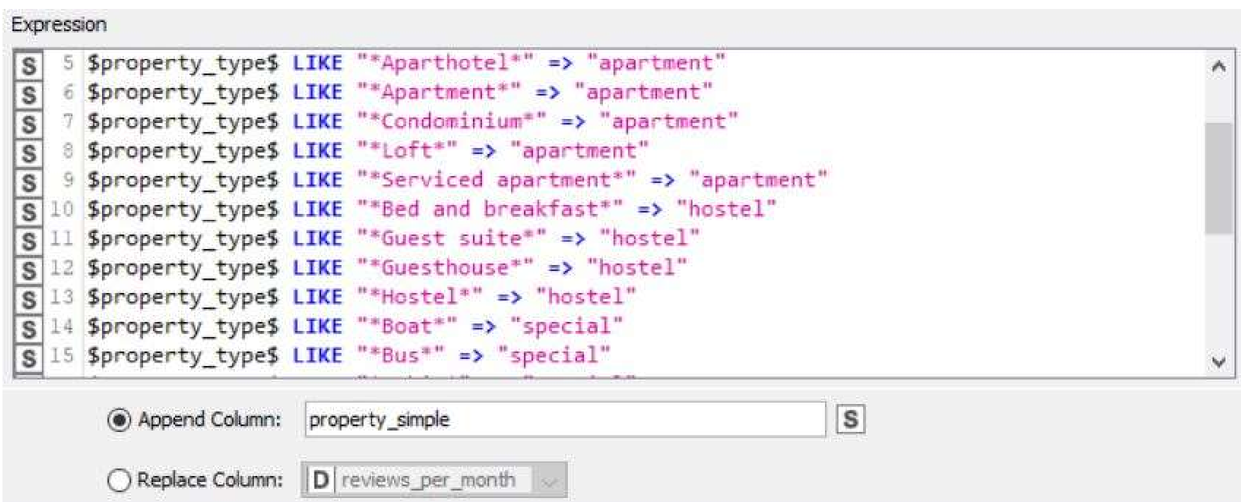
Q1a) Derive a new column called property_simple as follows (careful with capital letters and small letters)

Rule engine node to append "property_simple"



Linking a Rule Engine node will allow me to create certain rules and append the new column "Property_simple". The column "property_type" is selected and using the "LIKE" function, I can categorize the column and append the results into a new column name "property_simple". Below are the expressions used:

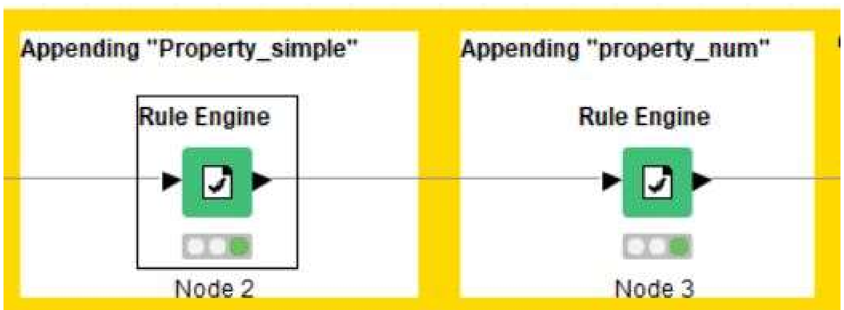


After executing this Rule Engine and opening up the Classified Values, I can see the new appended column "property_simple". Everything was in order and there were no "Other" values.

S property_type	S property_simple
Serviced apartment	apartment
Serviced apartment	apartment
Serviced apartment	apartment
Serviced apartment	apartment
Serviced apartment	apartment
Apartment	apartment
Serviced apartment	apartment
Serviced apartment	apartment
Serviced apartment	apartment
Serviced apartment	apartment
Apartment	apartment
Serviced apartment	apartment
Apartment	apartment
Apartment	apartment
Apartment	apartment
Apartment	apartment
Townhouse	house

b) Derive a new column (field) called property_num.

Rule engine node to append "property_num"



I linked another Rule Engine node to my previous node. This time, I will be selecting the newly appended "property_simple" column. I used the same "LIKE" function to create expressions for the new "property_num" column.

Expression

```
// TRUE => "default outcome"
$property_simple$ LIKE "*special*" => 1
$property_simple$ LIKE "*hostel*" => 2
$property_simple$ LIKE "*hotel*" => 3
$property_simple$ LIKE "*apartment*" => 4
$property_simple$ LIKE "*house*" => 5
TRUE => 0
```

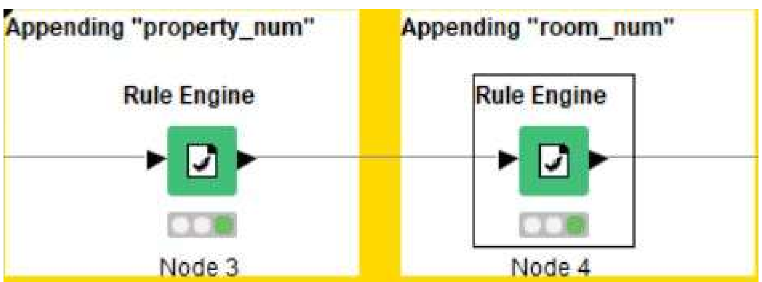
Append Column: property_num

Replace Column: S property_simple

The results are as shown below. Everything was in order and there was no rows with "0" values.

S property_simple	I property_num
house	5
house	5
house	5
house	5
house	5
apartment	4
house	5
special	1
house	5
house	5
apartment	4
apartment	4
apartment	4
apartment	4
apartment	4

c) Derive a new column called room_num.



To create the new column "room_num", I linked another rule engine to the previous rule engine. This time, I will be selecting the "room_type" column and using the same "LIKE" functions.

Below are the expressions I used for the new column.

Expression

```
// $string column name$ LIKE "*blue*" => "small and blue"
// TRUE => "default outcome"
$room_type$ LIKE "*Shared room*" => 1
$room_type$ LIKE "*Private room*" => 2
$room_type$ LIKE "*Entire home/apt*" => 3
TRUE => 0
```

(•) Append Column: room_num I

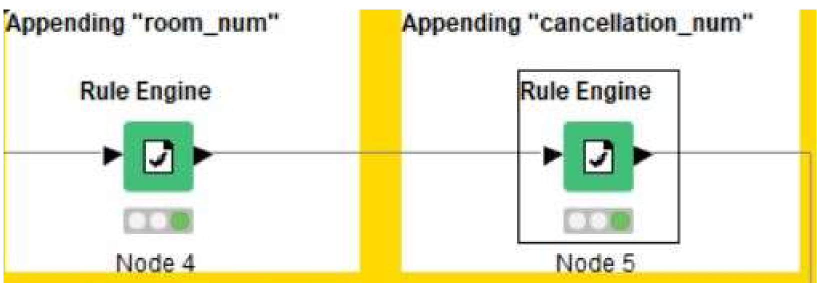
( ) Replace Column: I property_num

Executing the Rule engine node and opening the classified values shows me the new appended column "room_num" as shown below

room_type	room_num
Private room	2
Private room	2
Private room	2
Private room	2
Private room	2
Private room	2
Private room	2
Private room	2
Private room	2
Private room	2
Private room	2
Entire home/apt	3
Private room	2
Entire home/apt	3

d) Derive a new column called cancellation_num

Rule engine node to append "cancellation_num" column



Linking another rule engine node to append the new column "cancellation_num".

To create "cancellation_num", I will select "cancellation_policy" and use the same "LIKE" function.

Below are the expressions used in the rule engine

Expression

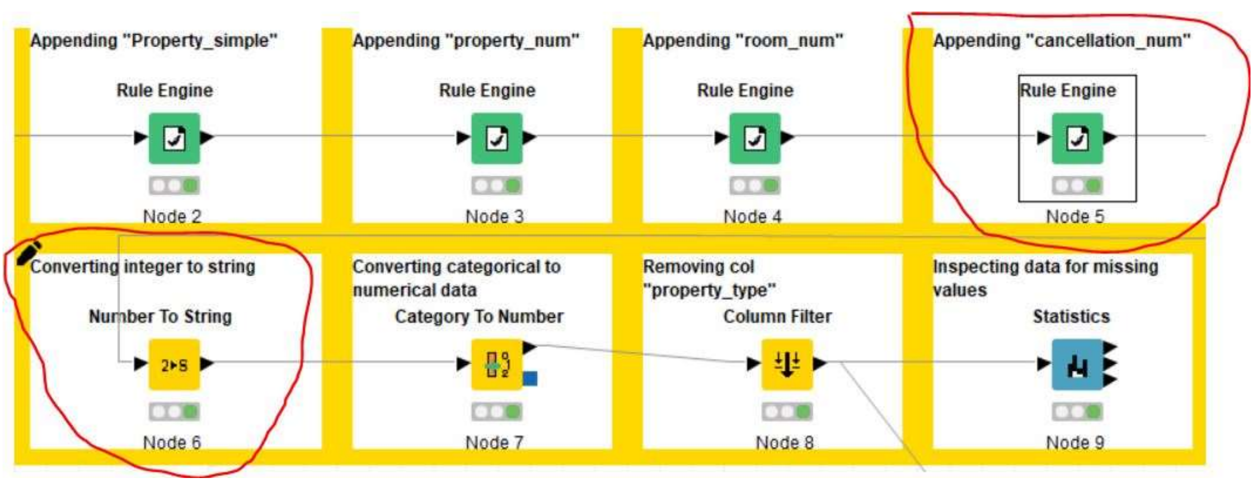
```
// TRUE => "default outcome"
$cancellation_policy$ LIKE "*super_strict_30*" => 1
$cancellation_policy$ LIKE "*strict_14_with_grace_period*" => 2
$cancellation_policy$ LIKE "*moderate*" => 3
$cancellation_policy$ LIKE "*flexible*" => 4
TRUE => 0
```

Executing the Rule engine node and opening the classified values shows me the new appended column "cancellation_num" as shown below

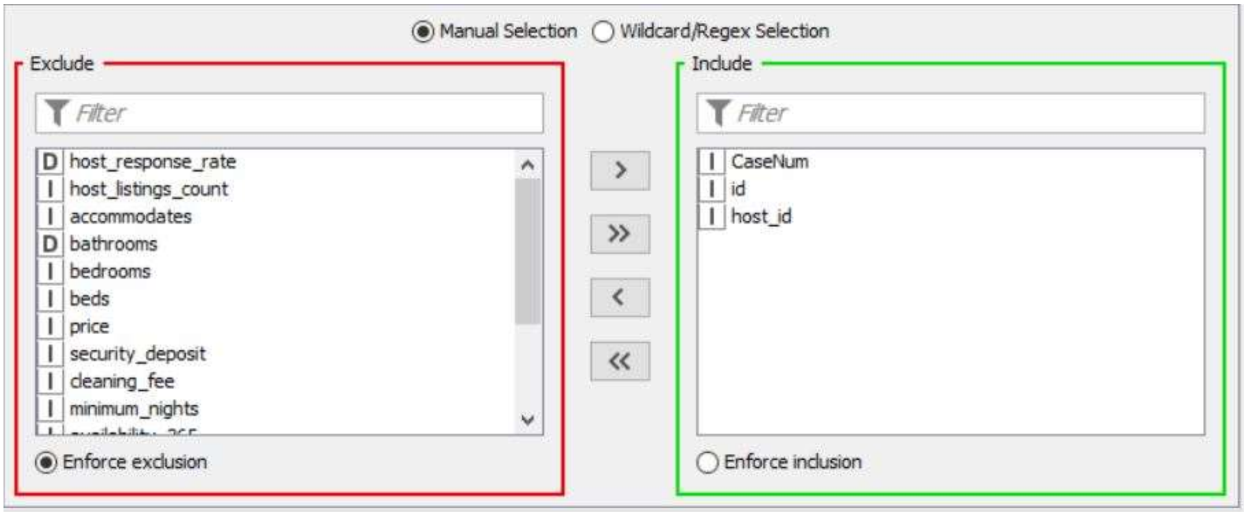
S cancellation_policy	I cancellation_num
moderate	3
moderate	3
moderate	3
strict_14_with_grace_...	2
moderate	3
moderate	3
moderate	3
moderate	3
moderate	3
moderate	3
strict_14_with_grace_...	2
strict_14_with_grace_...	2
moderate	3
moderate	3
moderate	3
moderate	3
flexible	4

Q2a) Convert the columns CaseNum, id and host_id to string type.

Number To String node to integer type columns to string type



Linking a “Number To String” node, will allow me to change the integer type columns “CaseNum”, “id” and “host_id” to a string type column. In the node, I will include only “CaseNum”, “id” and “host_id” as required by the question.



Executing the node and looking at the transformed input, I can now see that "CaseNum", "id" and "host_id" are now string type data as shown below.

Columns: 29	Column Type
CaseNum	String
id	String
host_id	String

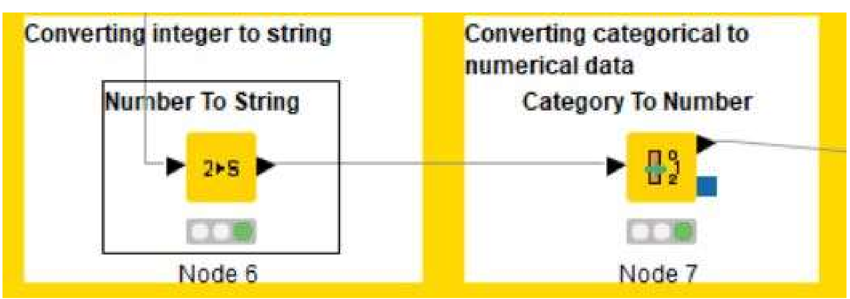
b) Convert the column host_is_superhost from categorical data to numeric data.

Using Category to Number node to change categorical data to numeric data

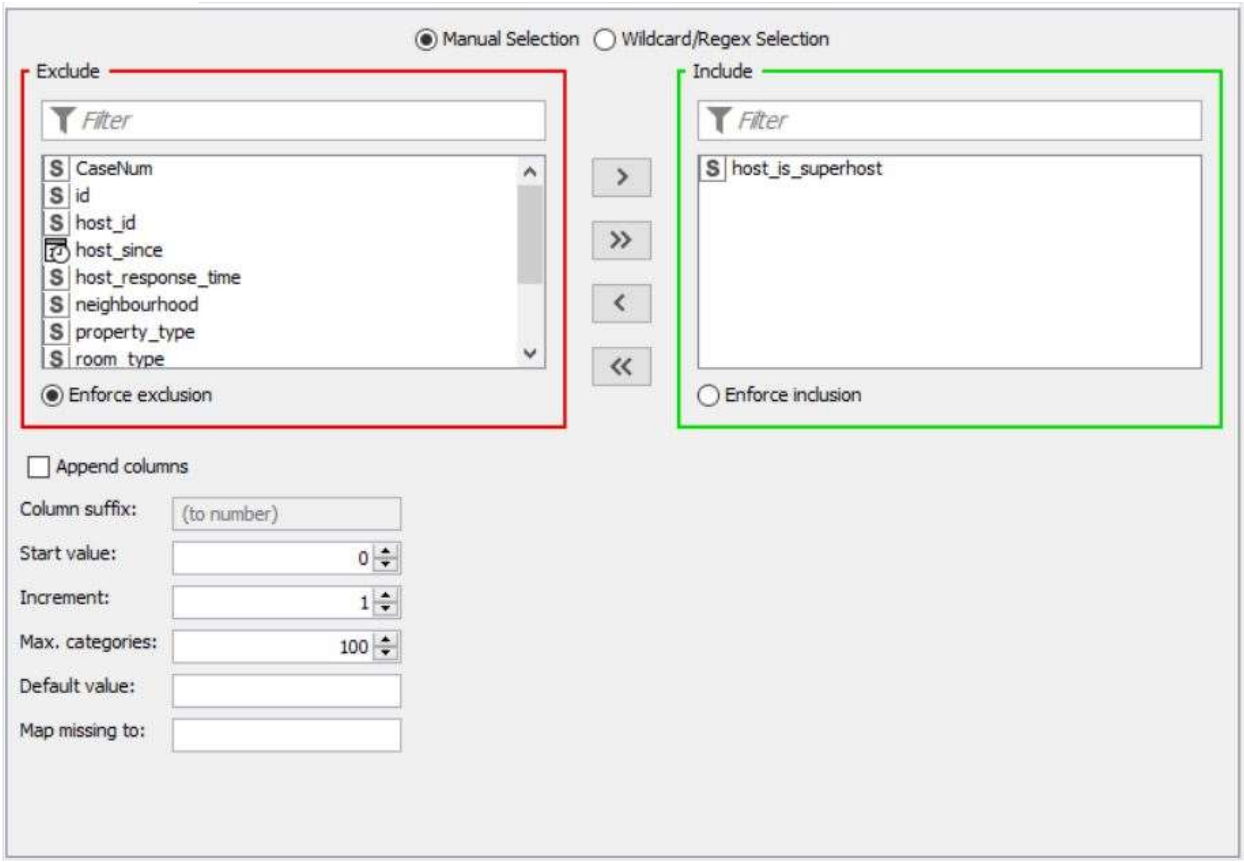
Taking a look at values at the data before I do any changes, I can see that the column "host_is_superhost" is a string data type and has only two values "f" and "t".

host_is_superhost	String	6					?	?	f	t

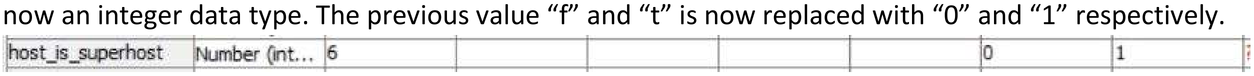
To change "host_is_superhost" column to a numeric data, I will link a "Category to Number" node.



In the node itself, I will include only "host_is_superhost" column and ensure that "Append columns" is unticked. I will leave the other settings untouched as I am okay with it.



Executing the node and looking at the processed data, I can now see that "host_is_superhost" column is now an integer data type. The previous value "f" and "t" is now replaced with "0" and "1" respectively.

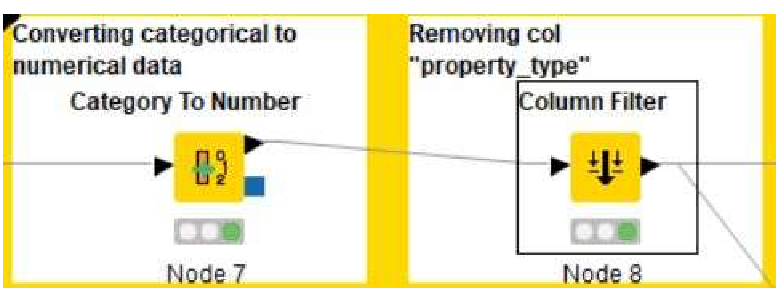


c) Remove the column (field) property_type. How many columns are left?

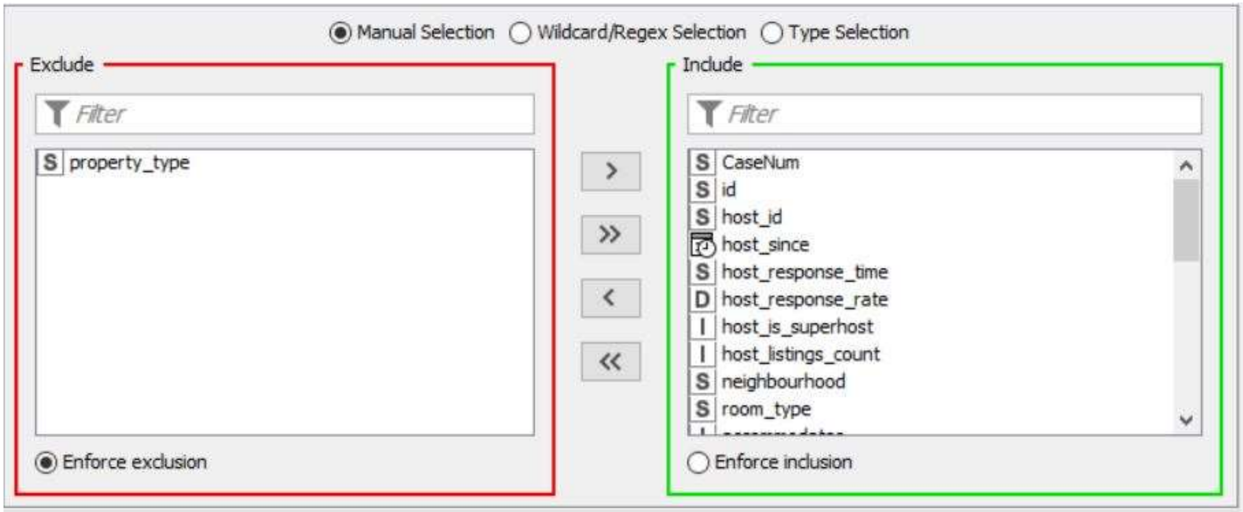
Using Column Filter node to remove unwanted columns

Looking at the data from "Category To Number" node, I can see that there are 29 columns.

I will now link the Column Filter node as it will allow me to remove specific columns.



In the column filter node, I will include every column and exclude "property_type" column as requested by the question.



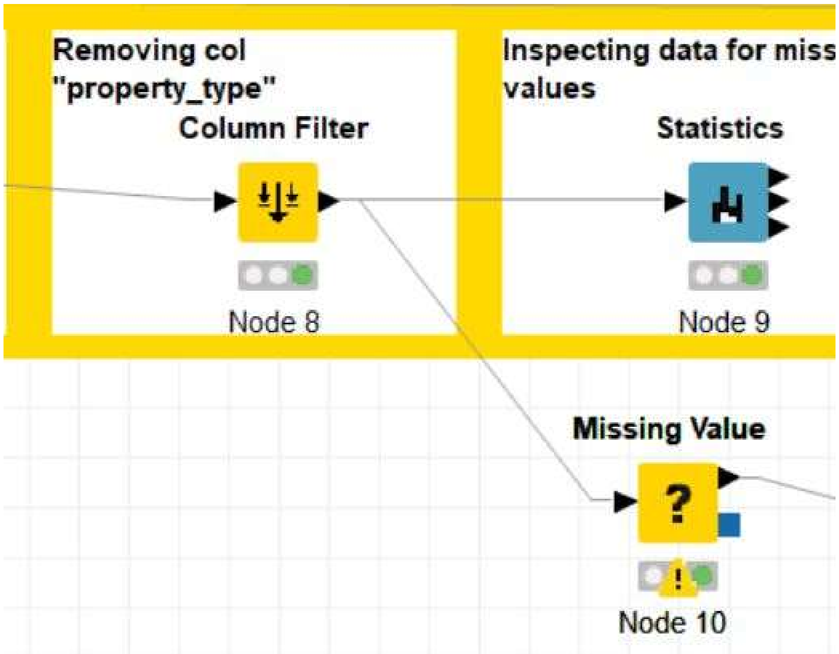
Executing the node and looking at the filtered table, I can now see that "property_type" column is now removed leaving the data with 28 columns as shown below.

Table "default" - Rows: 3148 Spec - Columns: 28

Columns: 28	Column Type	Column Index
CaseNum	String	0
id	String	1
host_id	String	2
host_since	Local Date T...	3
host_response_time	String	4
host_response_rate	Number (do...	5
host_is_superhost	Number (int...	6
host_listings_count	Number (int...	7
neighbourhood	String	8
room_type	String	9
accommodates	Number (int...	10
bathrooms	Number (do...	11
bedrooms	Number (int...	12
beds	Number (int...	13
bed_type	String	14
price	Number (int...	15
security_deposit	Number (int...	16
cleaning_fee	Number (int...	17
cancellation_policy	String	18
minimum_nights	Number (int...	19
availability_365	Number (int...	20
region	String	21
review_scores_rating	Number (int...	22
reviews_per_month	Number (do...	23
property_simple	String	24
property_num	Number (int...	25
room_num	Number (int...	26
cancellation_num	Number (int...	27

Question 2 d) There are some missing values in the data. Decide what you want to do for each case. Execute the handling of missing values. Justify why you chose the method for each case.

Statistics node to find out missing values



Connecting the statistics node and opening "Statistics View", I can now inspect which categories have missing data. Below are the results,

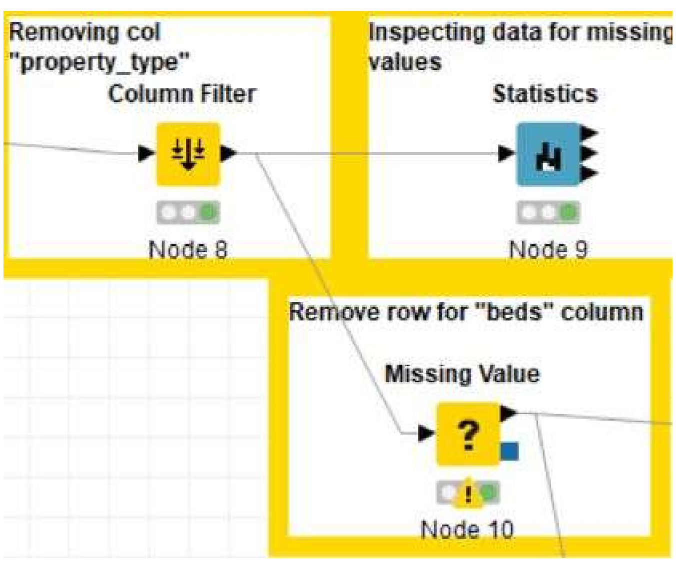
beds	0.0	2.4604	?	36	3.0111	4.8507	34.1659	1	0	0
security_deposit	0.0	207.5282	?	6,250	269.9482	8.6623	156.2953	791	0	0
cleaning_fee	0.0	37.9192	?	300	34.6388	1.5021	4.2374	697	0	0

Using the Statistics node, I have now found that the categories "beds", "security_deposit" and "cleaning_fee" are missing 1, 791 and 697 values respectively.

I now know which categories I should focus and fix.

Column	Action taken	Reason
beds	Remove row.	As the column "beds" does not have many missing values (1 row), I feel that removing the respective row itself is a reasonable method as there is already a large enough sample size of 3147 rows after omitting the row.
security_deposit	Multiple Imputation by Classification and Regression Trees in RStudio (CART)	At first, looking at the data, there were a lot of missing values and I felt that using the methods available in the "Missing Value" node would create a bias. I researched and found out about Multiple Imputation could be a suitable approach for my dataset. In RStudio, using the method CART method, I hope to reduce bias by drawing real values sampled from the data.
cleaning_fee	Multiple Imputation by Classification and Regression Trees in RStudio (CART)	Same reason as "security_deposit" column

1)**Missing Value node to remove row in "beds" column**

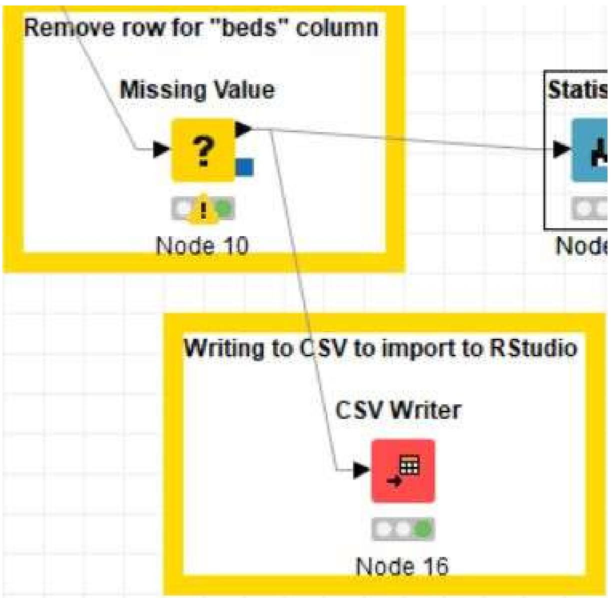


In the Missing Value node, I can choose different ways on how I can handle missing values in the data. Under the Missing Value Handler Selection, I will select remove row.

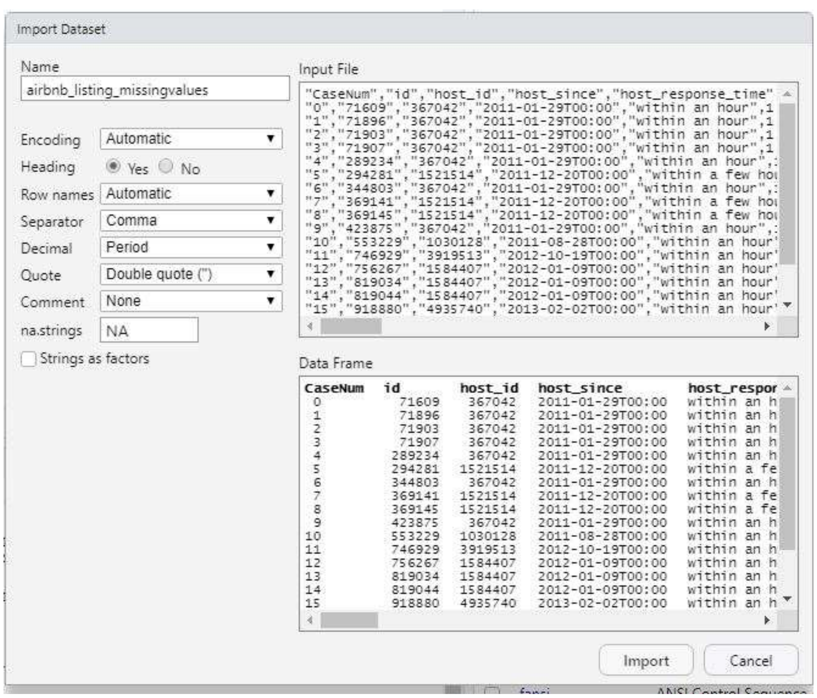
Attaching a Statistics node and opening up the statistics view shows me that there are no more missing values in "beds" column.

2) **Using Multiple Imputation by Classification and Regression Trees in RStudio (CART) to fix missing values of both "security_deposit" and "cleaning_fee" column.**

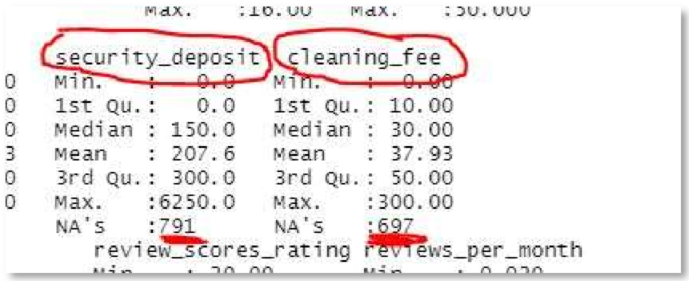
Firstly, I will connect a CSV Writer node to export the latest changed dataset as "airbnb_listing_missingvalues".



After that, I will open up RStudio and import the dataset as shown below.



In the console, using the command “summary(airbnb_listing_missingvalues)” I can verify the number of missing values and also view other information as well. The number of missing value is indicated as “NA’s”. “security_deposit” and “cleaning_fee” column has 791 and 697 missing vales respectively which is correct.



```
  security_deposit  cleaning_fee
0  Min.    :   0.0  Min.    :  0.00
0  1st Qu.:    0.0  1st Qu.: 10.00
0  Median :  150.0  Median : 30.00
3  Mean   :  207.6  Mean   : 37.93
0  3rd Qu.:  300.0  3rd Qu.: 50.00
0  Max.   :6250.0   Max.   :300.00
   NA's   :791      NA's   :697
     review_scores_rating reviews_per_month
```

3)Using CART method to handle missing values and generating multiple imputations

Next I will use the command:

my_imp = mice(airbnb_listing_missingvalues, m=5, method = "cart",maxit = 50, seed = 500).

- MICE stands for Multivariate Imputation via Chained Equations which is a package in RStudio that will allow me to use the CART method.
- “m=5” is the amount of imputations it will generate.
- “maxit=50” is the max amount of iterations that it takes, the more iterations the programme takes, the more accurate the prediction may be.
- “seed = 500” is so that the results would be reproducible.

The next commands would be

1) summary(airbnb_listing_missingvalues$security_deposit) & my_impimpsecurity_deposit
2) summary(airbnb_listing_missingvalues$cleaning_fee) & my_imp$imp$cleaning_fee

Doing “summary(airbnb_listing_missingvalues$security_deposit)“ command, will be able to show me information about “security_deposit” column such as its median, mean, max and missing values. I would also repeat this command for “cleaning_fee” column.

The median value is what I will be focused on as it will help me decide which imputation to choose out of the 5 generated later on.

To show the generated imputation I will use the command “my_impimpsecurity_deposit” and “my_impimpcleaning_fee”. This will show me 5 imputations with the missing values replaced as shown below for each variable.

After looking through the 5 imputations, I decided to take the 5th imputation for both columns as many of the replaced missing values are closer to the mean value as compared to the other imputations.

1.security_deposit imputations

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
    0.0     0.0   150.0   207.6   300.0  6250.0     791 
> my_imp$imp$security_deposit
       1    2    3    4    5
22   250  500  139  250  139
23     0  200  250    0  138
24   150  250  139  500  500
26   300  600  300 1600 6250
30   600 1600    0    0    0
35  6250 1600 6250  300  300
50   200  150  300    0  150
52     0    0    0  138    0
55     0    0    0    0    0
57     0    0    0    0    0
60     0  350  300    0  417
64   138  138    0    0  138
88  1000  200 1000 1000  500
96     0  450    0    0  200
97     0    0    0    0    0
106    0    0    0    0    0
107    0  200  138  150  200
109    0    0    0    0    0
110    0  200    0    0  137
113    0    0    0    0    0
116    0  150  150  150    0
120  200  150  150    0    0
133  400  400  150  138    0
134  450    0  135  400  450
135  400  450  150  400  400
136  150  200    0  200  200
140  500  200  150    0  150
143  400    0  150  138    0
144  200    0  150  150    0
148  200  200    0  150  200
150  140  150  138  140    0
152  150  200  150    0  150
156  150  200  150  140    0
157  150  200  150  200    0
158  200  200  150  150    0
168  400  400  200  400  450
169  150  200  150  150  150
170  150    0  150  150  150
172    0    0    0    0    0
176  150  200  150  150  150
177    0  200  150  140    0
183  280  200  300  350  200
185  200  150  150  150  200
198    0  150    0  206  150
206    0    0    0    0  150
214    0    0  200  200    0
219  150  150    0  141    0
222  200  150    0    0  150
```

2.cleaning_fee imputations

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
   0.00   10.00   30.00   37.93   50.00  300.00     697 
> my_imp$imp$cleaning_fee
     1   2  3   4   5
13  49  50 40  35  35
14  46  50 60  40  90
15  35  75 50  45  45
20  39  30 15  29  25
22  50  50 30 150   0
23   0  30 80   0   8
26  49  50 30  30  69
28  35  30 75  35  30
30  75  30  0   0   0
50  50  10 25  15  10
52   0   0  0   0   0
55   0   0  0   0   0
57   0   0  0   0   0
64  20  28 20   0   0
68   0   0  0   0   0
69   0   0  0   0   0
75   0   0  0   0   0
76   0   0  0   0   0
80   0   0  0   0   0
91   0   0  0   0   0
95   0   0  0   0   0
96   0  30 10   0   0
97  10   0  0   0   0
98   0   0  0   0   0
100  0   0  0   0   0
101  0   0  0   0   0
105 30  20 20  20  25
106  0   0  0   0   0
107 10  20 15  45  30
109  0   0  0   0   0
111  0   0  0   0   0
113  0   0  7   0   0
115  0   0  0   0   0
116  0  55 55  45   0
117  0   0  0   0   0
131  0   0  0   0   0
133 50  45 50  10   0
134 50   0 10  50  70
135 50  90 50  45  70
140 80  20 20  10  15
141 30  20 20  30   0
143 40   0 50   0   0
145  0   0  0   0   0
150 10  55  0   0   0
151  0   0  0   0   0
152 50  65 50   7  45
154  0   0  0  40   0
155  0   0  0  40   0
156 50  70 14   0   0
157 50  75 45  45   0
158 75 100 50  45   0
```

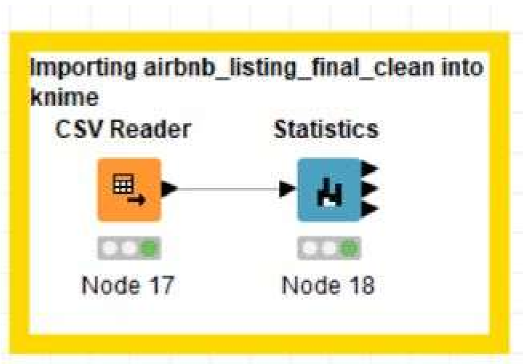
4)Exporting final dataset into Knime

Therefore, now that I have decided which imputation I would like to take, I can export imputation back into a dataset. For that, I will use the command “airbnb_listing_final_clean = complete(my_imp,5)”.

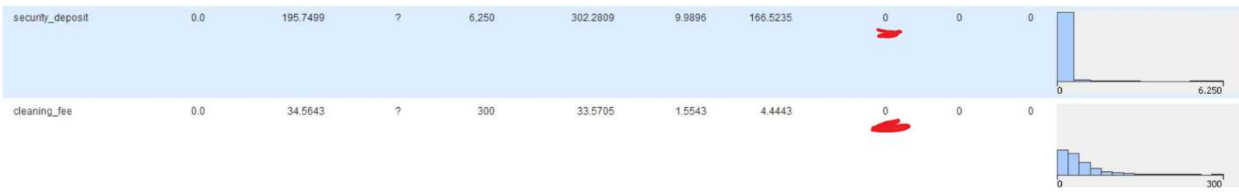
This will create a new dataset in RStudio named “airbnb_listing_final_clean”.

Next, I will export this final dateset into csv format by using this command: “write.csv(airbnb_listing_final_clean, file = "airbnb_listing_final_clean.csv", row.names = F)”.

And finally, to see the final dataset in Knime software, I will use the CSV reader node to open up the dataset in Knime and connect a Statistics node to ensure that there are no more missing values.



Opening up statistics view, I can verify that there are no more missing values for “security_deposit” and “cleaning_fee”.



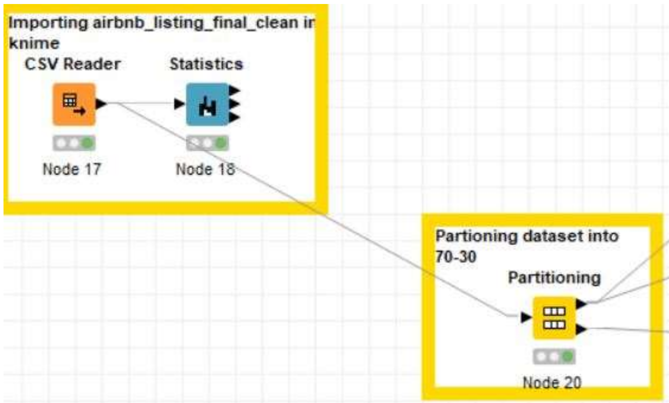
security_deposit	0.0	195.7499	?	6,250	302.2809	9.9896	166.5235	0	0	0
cleaning_fee	0.0	34.5643	?	300	33.5705	1.5543	4.4443	0	0	0

Question 3 (Option 2)

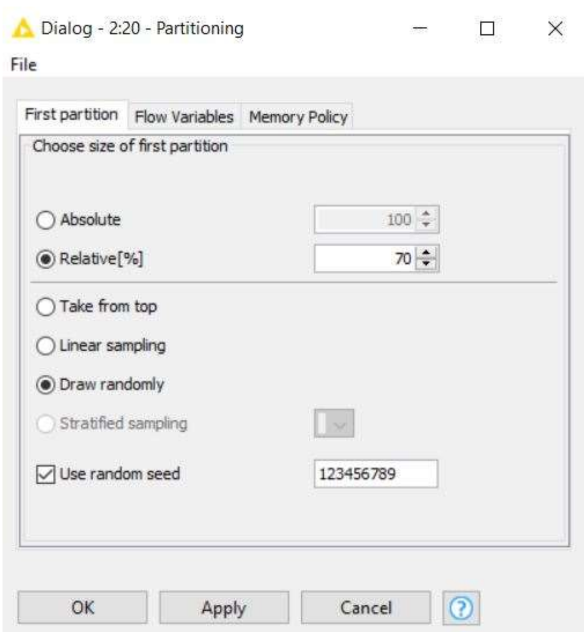
a) Perform linear regression using numeric property listing features with Price as the target. Do not use the features that are of string types. Write down the regression equation.

Partitioning dataset into training set(70% of data) and testing set(30% of data).

Before building a linear regression model, I will partition the dataset into 70-30 first.

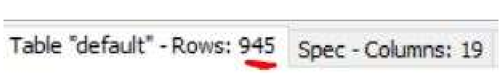


In the configuration menu, I will configure the size of the first partition to be 70% and use a seed of "123456789" so that the results can be replicable.



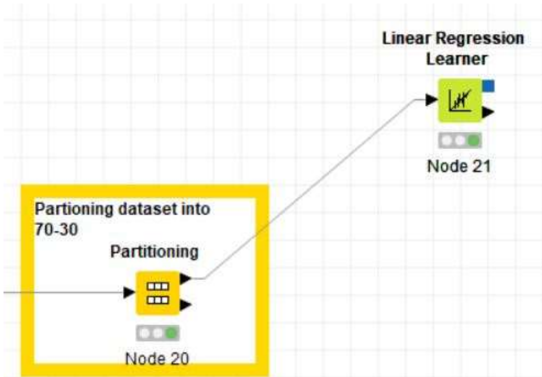
I can verify that my dataset has been partitioned as the first partition now contains 2202 rows

And the second partition now contains 945 rows.

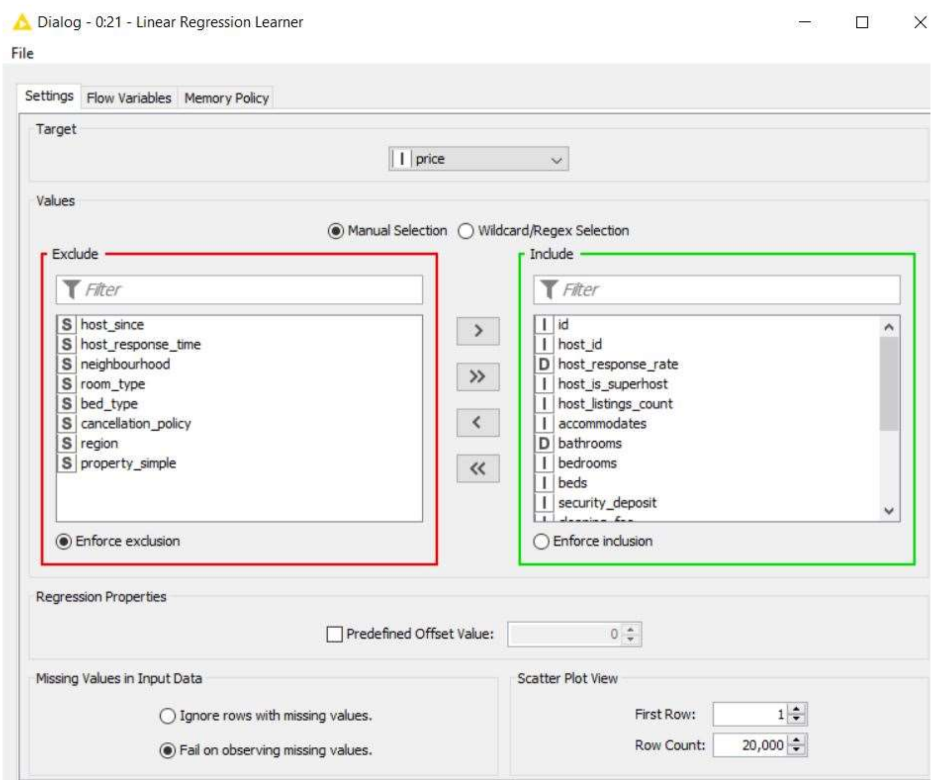


Training the Linear Regression Model using Linear Regression Learner node

Firstly I will connect a Linear Regression Learner node to my training dataset(70%) by connecting it to the top output part of my Partition node as such:



In the configuration menu of the Linear Regression Learner, I set the target (dependent variable) to "Price" as required by the question and only include numeric features.



Once done, I will execute it. Opening the Coefficient and Statistics output, I can determine which key features are important in predicting prices of listings by looking at the P value.

A predictor that has low p-value (< 0.05) is likely to be a meaningful addition to my regression model because changes in the predictor's value are related to changes in the target variable.

Table "Coefficients and Statistics" - Rows: 19 | Spec - Columns: 5 | Properties | Flow Variables

Row ID	S Variable	D Coeff.	D Std. Err.	D t-value	D P>|t|
Row1	id	0	0	2.781	0.005
Row2	host_id	-0	0	-0.351	0.726
Row3	host_response_rate	10.77	15.621	0.689	0.491
Row4	host_is_superhost	23.144	5.875	3.939	0
Row5	host_listings_count	-0.181	0.058	-3.118	0.002
Row6	accommodates	16.707	1.086	15.382	0
Row7	bathrooms	-5.583	1.776	-3.143	0.002
Row8	bedrooms	15.306	2.121	7.215	0
Row9	beds	-1.597	1.022	-1.563	0.118
Row10	security_deposit	0.017	0.008	2.097	0.036
Row11	cleaning_fee	0.378	0.082	4.631	0
Row12	minimum_nights	-8.716	2.062	-4.228	0
Row13	availability_365	0.08	0.018	4.488	0
Row14	review_scores_rating	0.695	0.188	3.7	0
Row15	reviews_per_month	-10.756	1.702	-6.321	0
Row16	property_num	-3.811	3.274	-1.164	0.245
Row17	room_num	71.725	4.657	15.403	0
Row18	cancellation_num	-0.983	3.454	-0.285	0.776
Row19	Intercept	-168.236	29.596	-5.684	0

From the table above, I identified 5 features that have a P value of above 0.05. The features are "host_id", "host_response_rate", "beds", "property_num" and "cancellation_num".

The data from these features is therefore not contributing much to the predictive model.

Excluding features one by one using backwards elimination

To make the prediction model more accurate, I will exclude features one by one, starting from the feature with the highest P values values.

After eliminating one by one I these are the features that are left with a P Value of less than 0.05.

Coefficients and Statistics - 0:21 - Linear Regression Learner

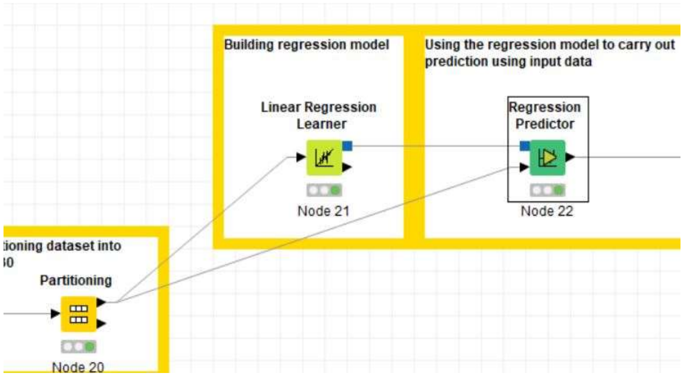
File Hilite Navigation View

Table "Coefficients and Statistics" - Rows: 14 | Spec - Columns: 5 | Properties | Flow Variables

Row ID	S Variable	D Coeff.	D Std. Err.	D t-value	D P>|t|
Row1	id	0	0	3.16	0.002
Row2	host_is_sup...	23.946	5.822	4.113	0
Row3	host_listings...	-0.167	0.056	-2.958	0.003
Row4	accommodates	15.695	0.815	19.254	0
Row5	bathrooms	-5.4	1.71	-3.158	0.002
Row6	bedrooms	15.012	2.074	7.238	0
Row7	security_de...	0.017	0.008	2.095	0.036
Row8	cleaning_fee	0.39	0.081	4.817	0
Row9	minimum_nig...	-9.034	2.024	-4.463	0
Row10	availability_...	0.08	0.018	4.54	0
Row11	review_scor...	0.709	0.187	3.789	0
Row12	reviews_per...	-10.816	1.675	-6.458	0
Row13	room_num	71.949	4.302	16.726	0
Row14	Intercept	-177.112	20.409	-8.678	0

Regression Predictor node to carry out prediction

After building a regression model in the Linear Regression Learner node, I will link a Regression Predictor node to carry out prediction using the regression model and training data from the Partitioning Node.



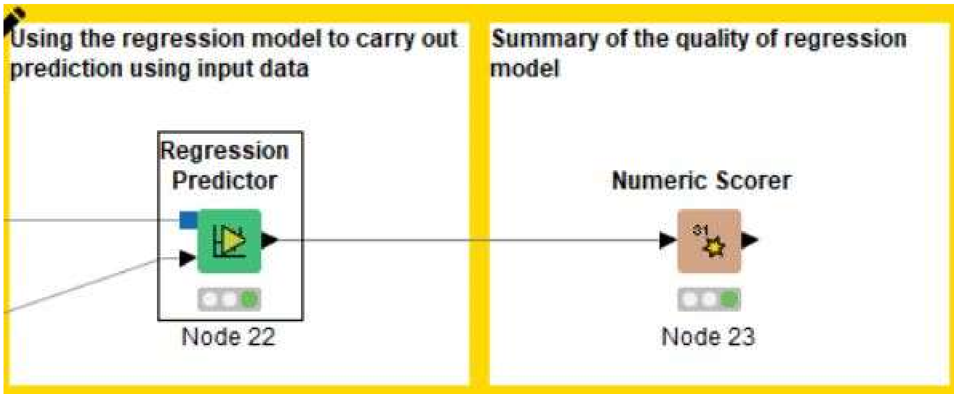
After executing the node and opening the data, I can see that a new column is created named “Prediction (Price)”. Putting “price” and “Prediction (price)” column side by side, I can compare and evaluate how accurate the regression model is. Below is a small part of the data:

I price	D Prediction (price)
94	112.432
104	115.654
208	160.816
417	230.609
49	73.02
60	86.263
60	79.293
104	110.868
53	103.611
165	106.365
150	110.545
65	110.407
76	64.423
69	51.318
140	180.059
49	53.362
72	79.909
50	-35.143
69	92.436
104	196.203

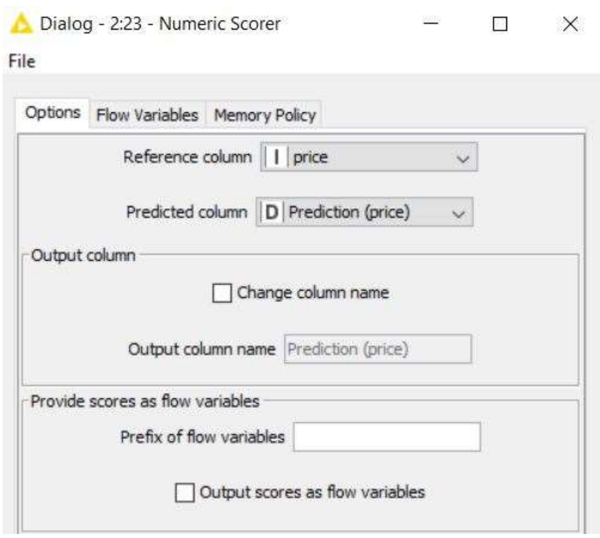
After assessing the data, many predicted values were close to the actual, but there were many that were far off as well.

Numeric scorer to assess Training Set

A Numeric Scorer node will allow me to assess the quality of my regression model better. Therefore, I will link it to the Regression Predictor node.



In the node I will set “price” as the reference column and “Prediction (price)” as the predicted column.



Executing the node and opening up the statistics displays various information for me to infer the quality of my regression model. Below are the results:

Row ID	D Prediction (price)
R^2	0.449
mean absolute error	51.569
mean squared error	9,966.359
root mean squared error	99.832
mean signed difference	-0
mean absolute percentage error	0.524

From what I have learnt, a regression model that can produce good prediction should have:

1) **High** R^2. Close to 1 as possible.
2) **Low** mean absolute error
3) **Low** mean square error
4) **Low** root mean squared error
5) **Low** mean signed error

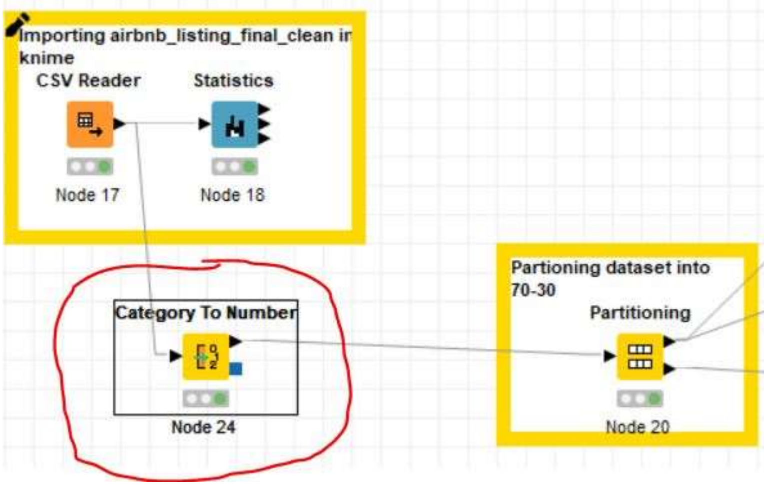
From the results above, I can infer that my current regression model can be improved as the R^2 value is far from 1.

Tuning my regression model

To make my regression model more accurate, I will:

1) Change string type features to numeric features using Category to Number node

I did this as I would like to see if these features would help in improving my regression model.



2) Exclude all features with high P values using backwards elimination
3) Try to raise the R^2 value by swapping in and out features in the Linear Regression Learner node

After experimenting for a while, the highest value R^2 I could achieve was 0.483. Below is the combination of features used to build the regression model. I double-checked to ensure that the P values of the features were less than 0.05.

Row ID	S Variable	D Coeff.	D Std. Err.	D t-value	D P>|t|
Row1	host_is_superhost	25.729	5.685	4.526	0
Row2	host_listings_count	-0.17	0.056	-3.049	0.002
Row3	accommodates	17.106	0.829	20.637	0
Row4	bathrooms	-6.178	1.734	-3.563	0
Row5	bedrooms	14.655	2.082	7.04	0
Row6	cleaning_fee	0.434	0.078	5.575	0
Row7	minimum_nights	-6.352	1.994	-3.186	0.001
Row8	availability_365	0.063	0.017	3.671	0
Row9	review_scores_rating	0.644	0.182	3.539	0
Row10	reviews_per_month	-9.647	1.652	-5.839	0
Row11	property_num	29.116	5.812	5.01	0
Row12	room_num	75.979	4.481	16.955	0
Row13	host_response_time (to number)	4.797	2.388	2.009	0.045
Row14	neighbourhood (to number)	2.145	0.315	6.814	0
Row15	room_type (to number)	-11.611	4.336	-2.678	0.007
Row16	cancellation_policy (to number)	15.076	4.007	3.763	0
Row17	region (to number)	-20.432	3.668	-5.57	0
Row18	property_simple (to number)	36.245	4.67	7.762	0
Row19	Intercept	-342.793	33.016	-10.383	0

Below is the summary of the regression model in the numeric scorer:

<u>Improved regression model</u>

Row ID	D Prediction (price)
R^2	0.483
mean absolute error	50.59
mean squared error	9,354.873
root mean squared error	96.721
mean signed difference	-0
mean absolute percentage error	0.535

<u>Previous regression model</u>

Row ID	D Prediction (price)
R^2	0.449
mean absolute error	51.569
mean squared error	9,966.359
root mean squared error	99.832
mean signed difference	-0
mean absolute percentage error	0.524

Overall, there was a slight improvement across the board. The R^2 value increased slightly, and the mean absolute and squared error decreased slightly as well.

Forming a regression equation

Since I have come up with a regression model, I can now try to form a regression equation:

Y = a +bX -> intercept value + (variable 1 x coeff.)

- ➔ Price = -327.609
  + (host_is_superhost * 25.729)

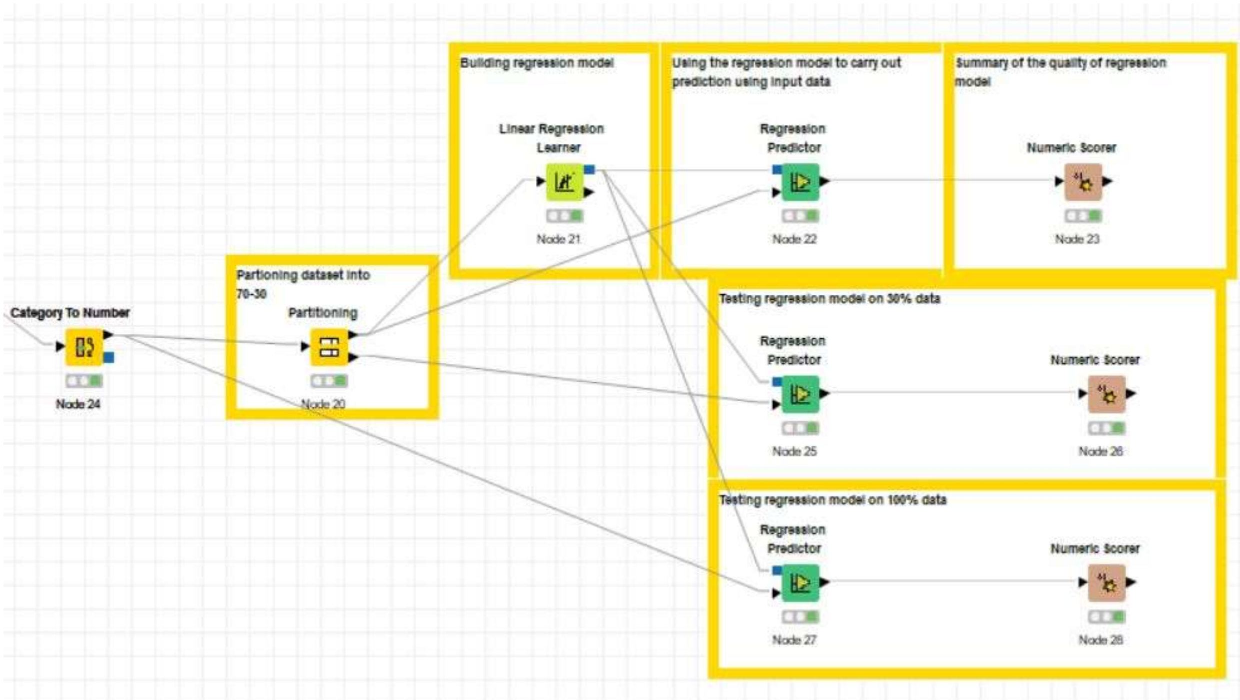
+ (host_listings_count * -0.17)
+ (accommodates * 17.106)
+ (bathrooms * -6.178)
+ (bedrooms * 14.655)
+ (cleaning_fee * 0.434)
+ (minimum_nights * -6.352)
+ (availability_365 * 0.063)
+ (review_scores_rating * 0.644)
+ (review_scores_month * -9.647)
+ (property_num * 29.116)
+ (room_num * 75.979)
+ [host_response_time(to number) * 4.797]
+ [neighbourhood(to number) * 2.145]
+ [room_type(to number) * -11.611]
+ [cancellation_policy (to number) * 15.076]
+ [region(to number) * -20.432]
+ [property_simple(to number) * 36.245]

b) Comment on the accuracy of your regression model. Which are the important features that can determine the price of the listing? Do you have any other findings?

Testing regression model on testing (30% data) and full data (100% data)

To further test out the accuracy of my regression model, I will firstly connect the regression model and 30% of the data from the Partitioning node to another pair of Regression Predictor and Numeric Scorer node.

I will also test the regression model on 100% of the data by getting the data from the Category to Number node.



Below is the summary of how the regression model performed in the 30% testing data and 100% full data:

<u>30% testing data</u>

Row ID	D Predicti...
R^2	0.59
mean absolute error	47.952
mean squared error	6,810.454
root mean squared error	82.525
mean signed difference	-2.731
mean absolute percentage error	0.517

<u>100% testing data</u>

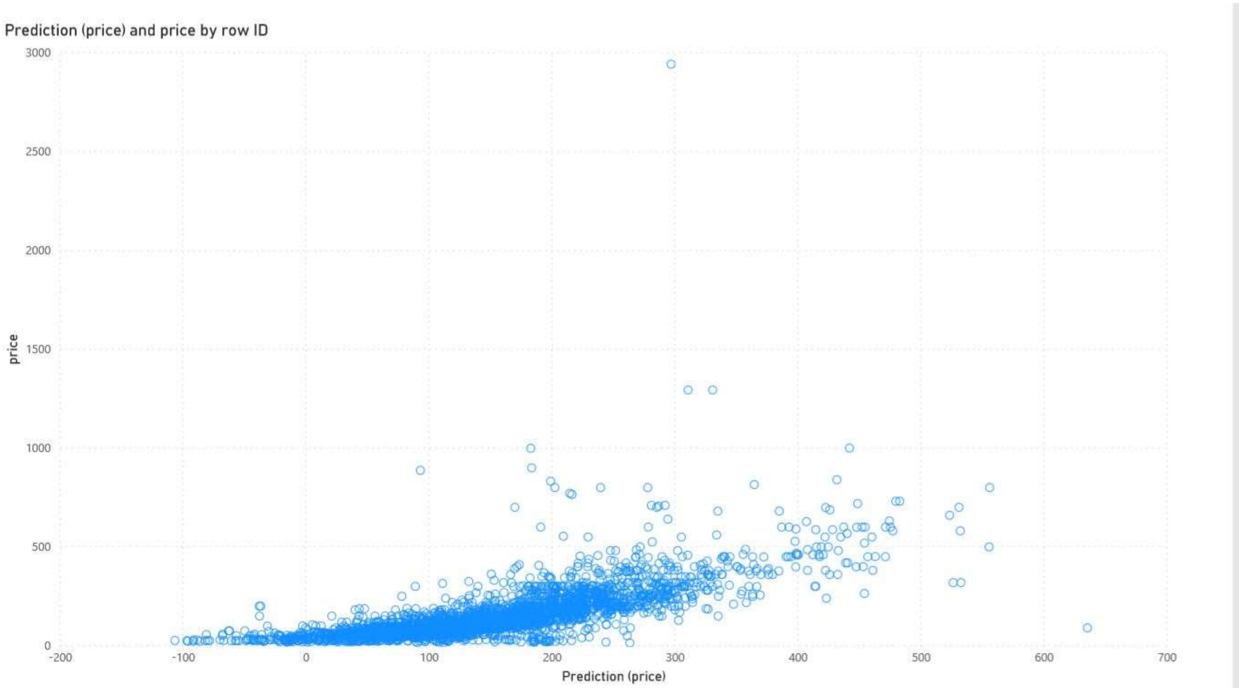
Row ID	D Prediction (price)
R^2	0.515
mean absolute error	49.871
mean squared error	8,565.61
root mean squared error	92.551
mean signed difference	-0.82
mean absolute percentage error	0.53

From the results, overall, there was a general slight improvement in the quality of the regression model when tested. The R^2 value in both testing and full data was higher of about 0.59 and 0.515 respectively as compared to 0.485 when tested using the 70% training data.

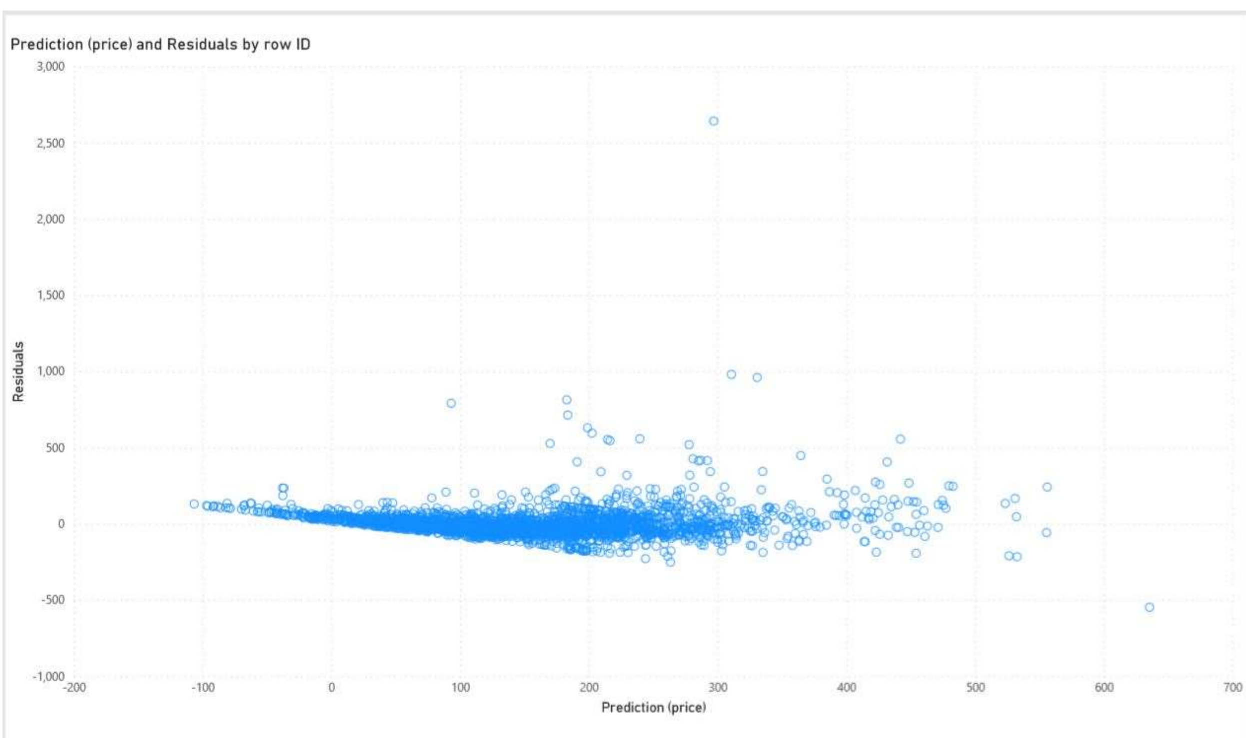
Other aspects across both summaries also has seen an improvement as their values are lower.

Using power BI to visualise the data

Prediction (price) and price by row ID



Prediction (price) and Residuals by row ID



From the two scatterplots that I have created in Power BI, I can infer that my regression model is not 100% accurate and can only predict the prices of listings to a limited extent.

In the first graph, there is a slight correlation between Prediction(price) and Price. However, as x-axis increase, the points start being scattered wider. Many points are also clustered in the beginning, which is probably because many listings are at $0-200 price range.

In the second graph, a straight line could be somewhat formed. Many points hover around the 0 on the y-axis. This tells me that the predictions made are close to the original data. However, when the x-axis increases, many points start to spread out from the 0 on the y-axis.

**RMSE for both testing datasets are above $80k+. In the context of predicting airbnb house prices, this would not be acceptable as the deviation from actual to predicted values is considered too far off. Therefore performance could be improved through feature engineering and feature selection.

Therefore, from the 2 graphs, I can infer that the regression model is only accurate in prediction to a limited extent.

Determining which features are important using Package “Boruta” in RStudio

<u>Description</u>

After some research, I have decided to use the Package “Boruta” in RStudio to determine feature importance.

Boruta is a feature selection algorithm. It tries to capture all the important, interesting features I might have in my dataset with respect to an outcome variable which is price of the listings in this case.

Firstly, it adds randomness to the given data set by creating shuffled copies of all features (which are called shadow features).

It then compares importance of attributes with importance of shadow attributes. Attributes that have significantly worst importance than shadow ones are being consecutively dropped and are known as Confirm Unimportant.

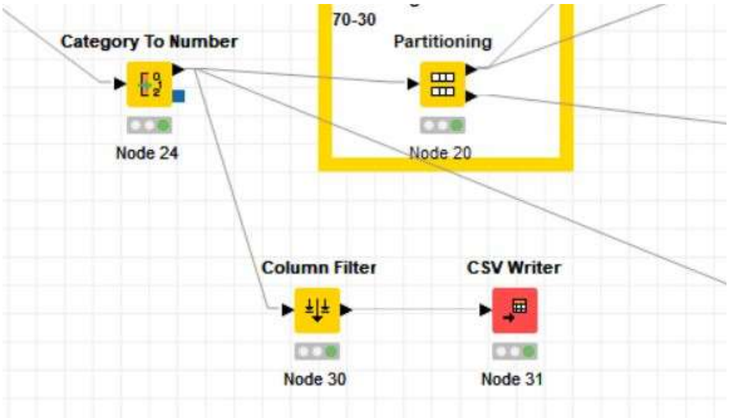
On the other hand, attributes that are significantly better than shadows are admitted as Confirmed important

some attributes may be left without a decision. They are claimed as Tentative.

Importing dataset into RStudio

Firstly, since my regression model only used numeric features, I will need to ensure that the data that I will be importing into RStudio only contains numeric features. Therefore, I will use a Column filter node to exclude string type features.

I will also connect a CSV writer downstream to export the data into a CSV format to be imported into RStudio.



I will name the dataset as “airbnb_listing_output2”.

After importing the dataset into RStudio, I will need to add the Boruta package

```
# Libraries
library(Boruta)
```

Once done I will specify the data and take a look at the structure of the data:

```
# Data
data(airbnb_listing_output2)
str(airbnb_listing_output2)
```

Below is the structure of the data where I can verify that my dataset is correct

```
> str(airbnb_listing_output2)
'data.frame':   3147 obs. of  26 variables:
 $ id                          : int  71609 71896 71903 71907 289234 294281 344803 369141 369145 423875 ...
 $ host_id                     : int  367042 367042 367042 367042 367042 1521514 367042 1521514 1521514 367042 ...
 $ host_response_rate          : num  1 1 1 1 1 0.86 1 0.86 0.86 1 ...
 $ host_is_superhost           : int  0 0 0 0 0 0 0 0 0 0 ...
 $ host_listings_count         : int  9 9 9 9 9 6 9 6 6 9 ...
 $ accommodates                : int  6 3 3 6 8 2 1 2 2 3 ...
 $ bathrooms                   : num  1 0.5 0.5 1 2 1 1 1 1 1 ...
 $ bedrooms                    : int  2 1 1 1 3 1 1 1 1 1 ...
 $ beds                        : int  3 1 2 7 1 1 1 1 1 2 ...
 $ price                       : int  206 94 104 208 417 65 49 60 60 104 ...
 $ security_deposit            : int  278 139 139 278 278 150 139 150 150 139 ...
 $ cleaning_fee                : int  56 28 28 69 83 41 28 42 42 28 ...
 $ minimum_nights              : int  1 1 1 1 2 2 2 2 2 1 ...
 $ availability_365            : int  353 355 346 172 239 336 357 340 349 300 ...
 $ review_scores_rating        : int  84 81 89 82 97 89 90 90 88 88 ...
 $ reviews_per_month           : num  0.15 0.22 0.38 0.25 0.14 1.35 0.5 1.17 1.75 0.19 ...
 $ property_num                : int  5 5 5 5 5 4 5 1 5 5 ...
 $ room_num                    : int  2 2 2 2 2 2 2 2 2 2 ...
 $ cancellation_num            : int  3 3 3 2 3 3 3 3 3 3 ...
 $ host_response_time..to.number. : int  0 0 0 0 0 1 0 1 1 0 ...
 $ neighbourhood..to.number.   : int  0 0 0 0 0 1 0 1 1 0 ...
 $ room_type..to.number.       : int  0 0 0 0 0 0 0 0 0 0 ...
 $ bed_type..to.number.        : int  0 0 0 0 0 0 0 0 0 0 ...
 $ cancellation_policy..to.number.: int  0 0 0 1 0 0 0 0 0 0 ...
 $ region..to.number.          : int  0 0 0 0 0 1 0 1 1 0 ...
 $ property_simple..to.number. : int  0 0 0 0 0 1 0 2 0 0 ...
```

After taking a look at the structure, I can now try to execute Boruta. I will use the following code:

```
> boruta <- Boruta(price ~ ., data = airbnb_listing_output2, doTrace =2)
```

In the code above, “Price” is set as the response variable and I included all variables by specifying “.”.

Once done, I will run the codes. Several iterations will be produced, and as it is being produced, outputs are being shown as well:

```
 12. run of importance source...
After 12 iterations, +25 secs:
 confirmed 23 attributes: accommodates, availability_365, bathrooms, bedrooms, beds and 18 more;
 rejected 1 attribute: bed_type..to.number.;
 still have 1 attribute left.

 13. run of importance source...
 14. run of importance source...
 15. run of importance source...
 16. run of importance source...
After 16 iterations, +33 secs:
 confirmed 1 attribute: host_response_time..to.number.;
 no more attributes left.

>
```

We can see that after 16 iterations, the programme has finished as all attributes have been either identified as Confirmed or Unconfirmed.

More iterations would be needed if an attribute is tentative or undecided.

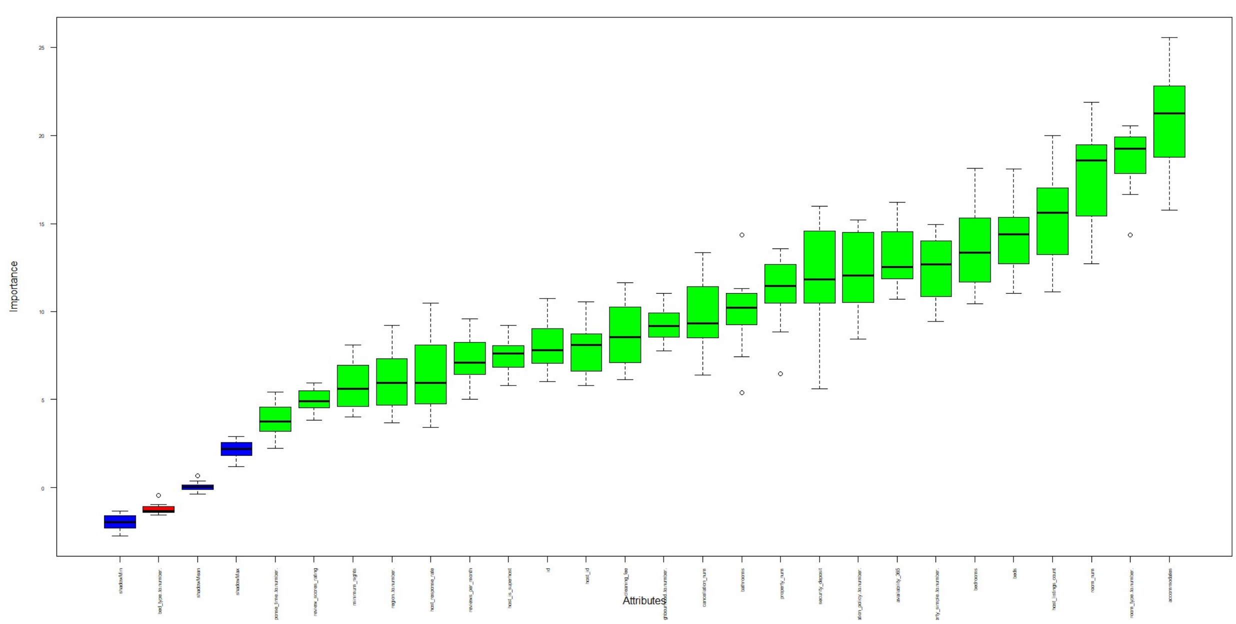
Viewing which features are most important

Using the code:

```
> print(boruta)
Boruta performed 16 iterations in 32.59203 secs.
 24 attributes confirmed important: accommodates, availability_365, bathrooms, bedrooms, beds and 19 more;
 1 attributes confirmed unimportant: bed_type..to.number.;
>
```

I can see feature importance by printing a box-plot diagram which illustrates to me the Confirmed important features and Confirmed Unimportant features

1) Green box-plots represent Confirmed Important features.
2) Red box-plots represent Confirmed Unimportant features.
3) Blue box-plots represent the minimum, maximum and average of Shadow Attributes.



As the feature names are small, I will list the features from left to right in accordance to the graph

shadowMin
bed_type (to number)
shadowMean
shadowMax
host_response_time (to number)
review_scores_rating
minimum_nights
region (to number)
host_response_rate
reviews_per_month
host_is_superhost
id
host_id
cleaning_fee
neighbourhood (to number)
cancellation_num
bedrooms
property_num
security_deposit
cancellation_policy (to number)
availability_365
property_simple (to number)
bedrooms
beds
host_listings_count
room_num
room_type (to number)
accomodates

From the box-plot diagram, I can conclude that:

The top 10 most important features that determine the price of the listing are

1) Accomodates
2) room_type(to number)
3) room_num
4) host_listings_count
5) beds
6) bedrooms
7) property_simple(to number)
8) availability_365
9) cancellation_policy(to number)
10) security deposit

The only unimportant features is “bed_type (to number)” in determining the price of listings.