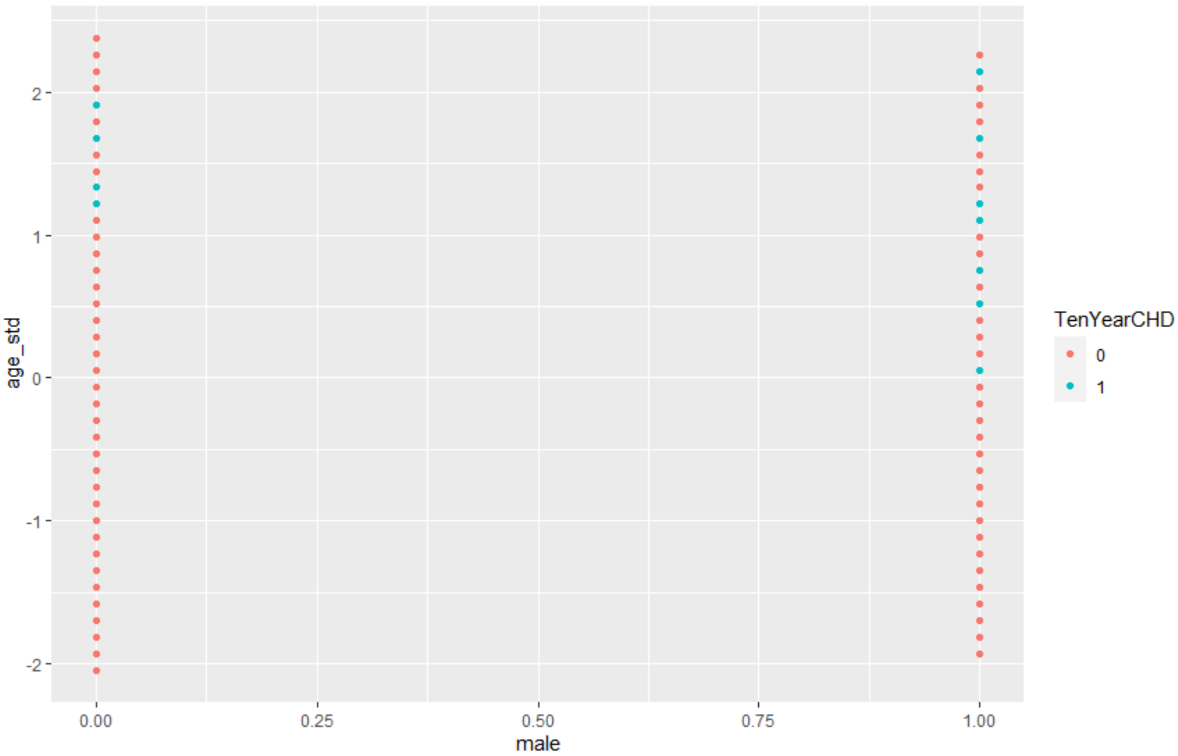
DTI5126[EG]: Fundamentals Data Science

Assignment 3
Clustering_Evaluation

Part A.1: Clustering → Kmeans

a) At first this is the distribution of the data, we can see how the 'cyan' color is very few, denoting that a 'Yes/1' value is less abundant/representative in the dataset than the 'No' value.

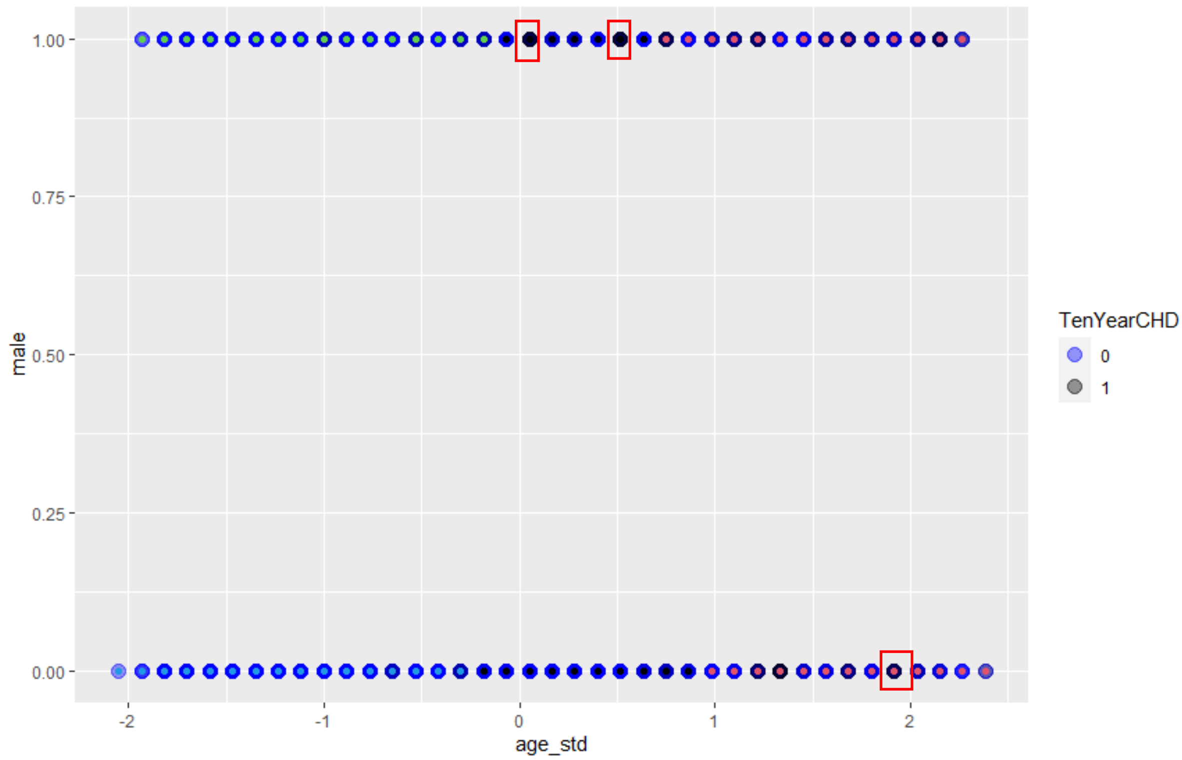


For K = 4 → We can see that the data is clustered among the 4 clusters where the 1st and 4th clusters contain the majority of the '0/No' whereas the 2nd clusters contains almost half the 'Yes/1' data.

	0	1
1	1039	202
2	720	294
3	825	96
4	1012	52

The inside cluster denotes the predicted cluster, whereas the outside outline denotes the True label. Obviously very few points denote the black outline (Yes), so it was highlighted manually to ease this

Cluster 1 → Blue, cluster 2 → Black, cluster 3 → Red, cluster 4 → Green

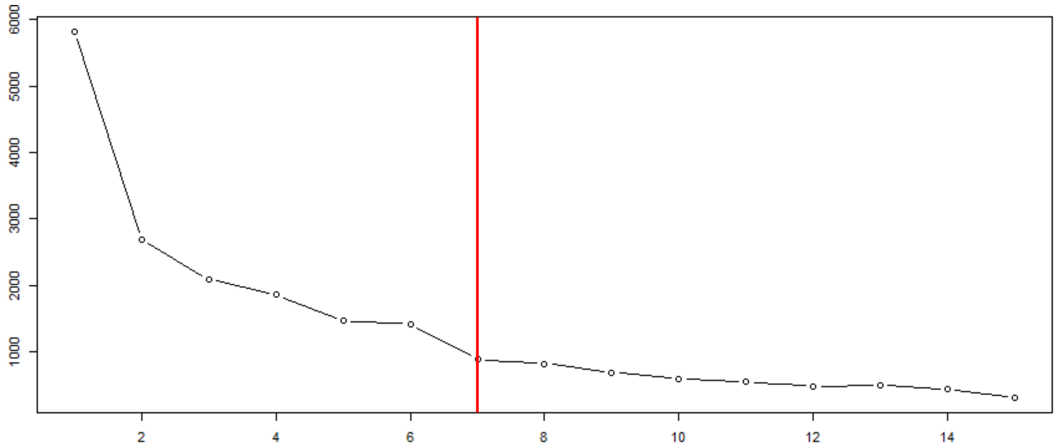


Here we can see that most of the blue outline (No) were grouped in the green and blue clusters (1 and 4), fewer in the red (3rd) cluster, while very few – compared to other—in the black fill (2nd cluster)

However, when we look at the black outline (highlighted by red squares), we can see that some were grouped in the black cluster (2nd) while fewer in the red (3rd) cluster.

This might need more investigation as the optimal output for each cluster would be very low distribution of one of the target values accompanied by a high distribution for the other value (something like cluster_4)

b) Elbow Method:

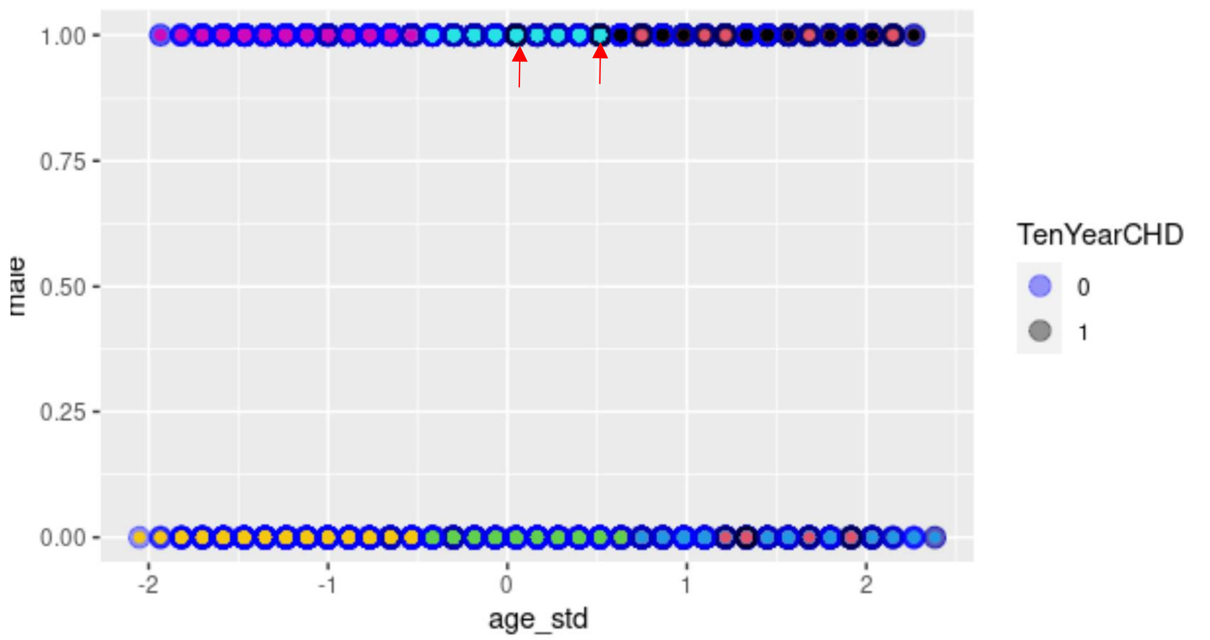


Here, we can notice that at point 7 (Red line) the reduction in variation (x-axis) starts to stabilize and this seems as the optimal point.
Thus, we choose K to be equal 7.

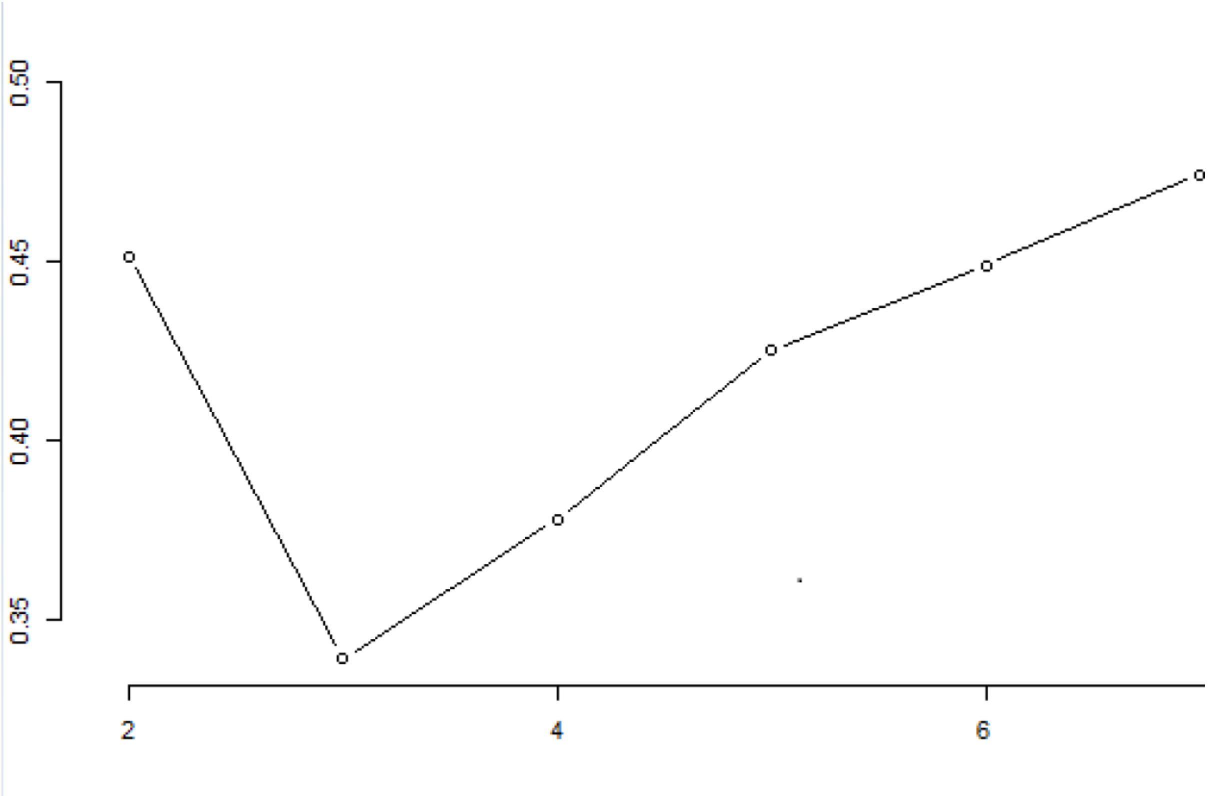
K = 7 →

	0	1
1	767	87
2	604	0
3	440	132
4	384	0
5	653	57
6	748	35
7	0	333

We can see that we're achieving more our objective. For example, cluster 2 and 4 has no values for the 'Yes/1' value, so these clusters are only for the 'No/0' records. Also, most of the 'Yes/1' values are represented in one cluster ALONE without any values for 'No/0' which is cluster 7.
The same colors for the 1st 4 clusters, cluster 5→ yellow, cluster 6→ white, cluster 7→ cyan
Here we can see that most of the black outline 'yes/1' are filled with cyan—cluster 7and none of them is filled with black or green (c2, c4)

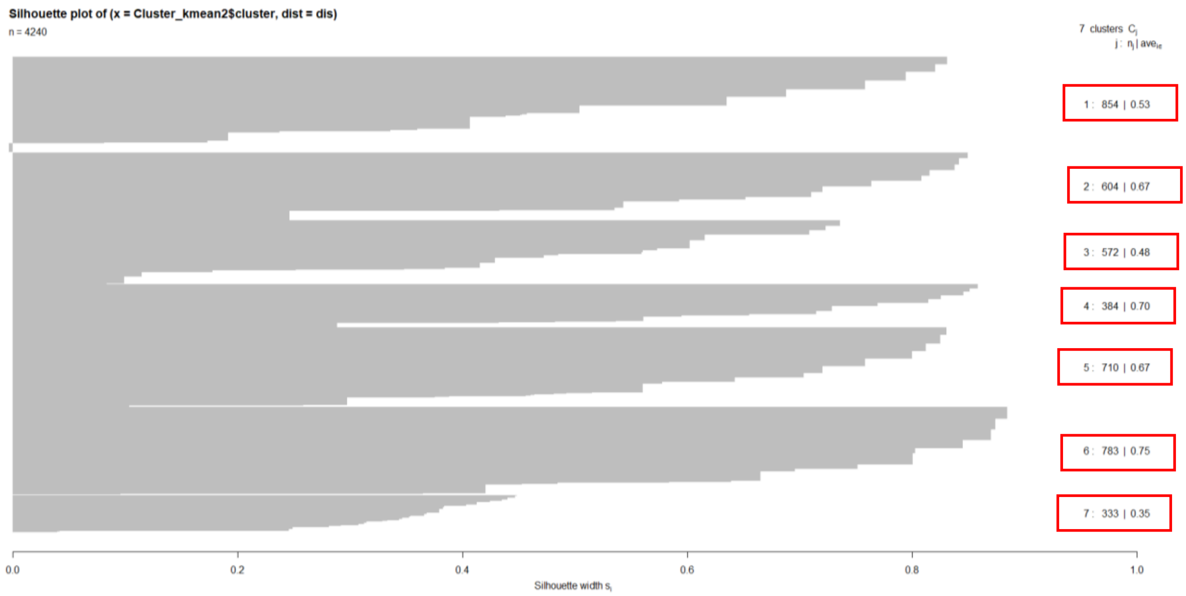


c) The silhouette coefficient for different Ks →



From 2Ks till out elbow K-d (7), we can find that 7 clusters had the highest silhouette score.

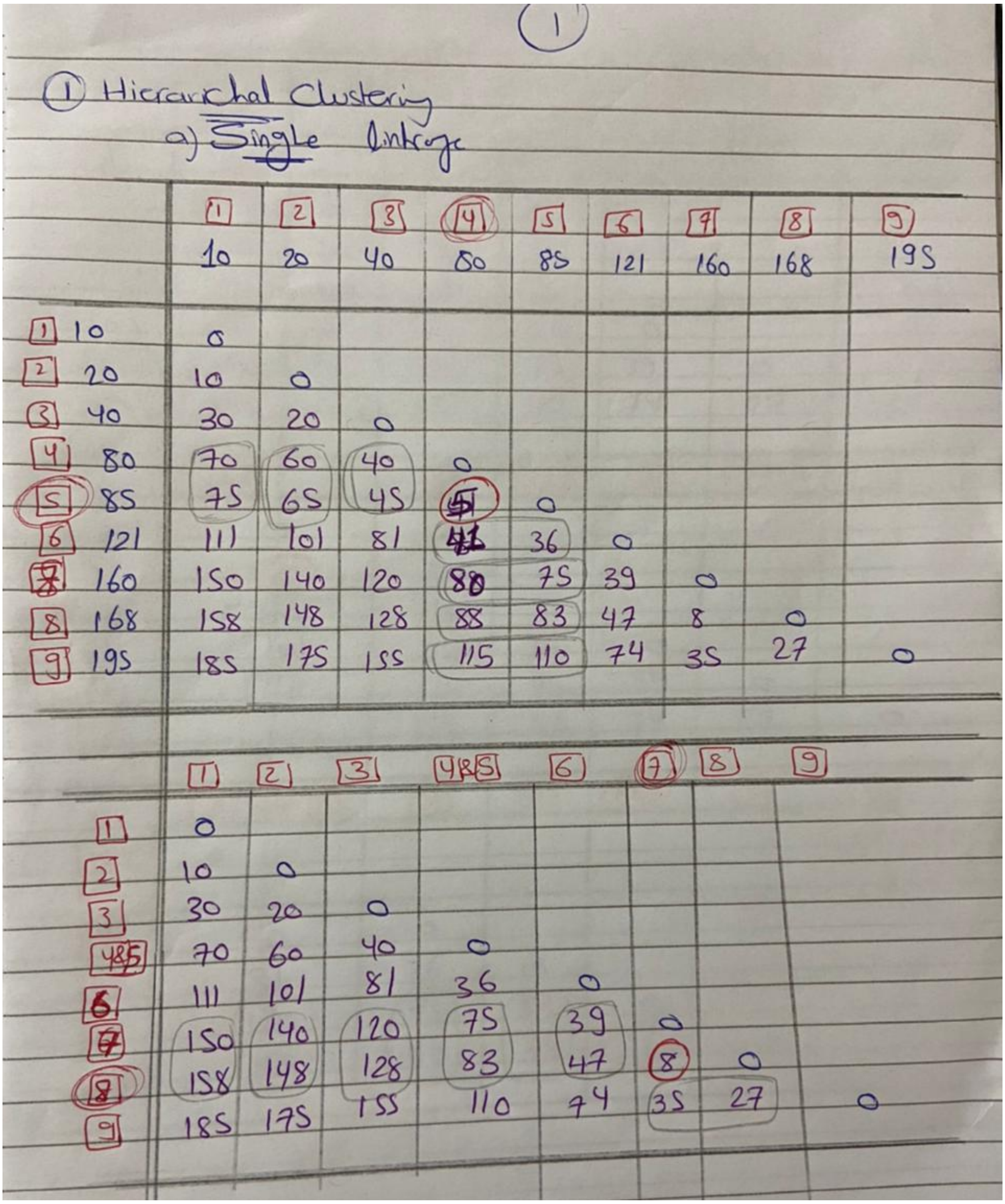
If we want to check for each cluster inside the 7 clusters: where the silhouette score is highest for the 6th and 4th clusters (70-75%)



Part A.2: Clustering → Hierarchal

The problem was solved by hand, however, dendogram was plotted via code to check the values which aligned together.

Single



1

① Hierarchal Clustering

a) Single linkage

	1 10	2 20	3 40	4 80	5 85	6 121	7 160	8 168	9 195
1 10	0								
2 20	10	0							
3 40	30	20	0						
4 80	70	60	40	0					
5 85	75	65	45	5	0				
6 121	111	101	81	41	36	0			
7 160	150	140	120	80	75	39	0		
8 168	158	148	128	88	83	47	8	0	
9 195	185	175	155	115	110	74	35	27	0

	1	2	3	4&5	6	7	8	9
1	0							
2	10	0						
3	30	20	0					
4&5	70	60	40	0				
6	111	101	81	36	0			
7	150	140	120	75	39	0		
8	158	148	128	83	47	8	0	
9	185	175	155	110	74	35	27	0

	1	2	3	4&5	6	7&8	9
1	0						
2	10	0					
3	30	20	0				
4&5	70	60	40	0			
6	111	101	81	36	0		
7&8	150	140	120	75	39	0	
9	185	175	155	110	74	27	0

	1&2	3	4&5	6	7&8	9
1&2	0					
3	20	0				
4&5	60	40	0			
6	101	81	36	0		
7&8	140	120	75	39	0	
9	175	155	110	74	27	0

	x and 3	y	6	z	9
x&3	0				
~~3~~					
y	40	0			
6	81	36	0		
z	120	75	39	0	
9	155	110	74	27	0

let
- x = 1&2
- y = 4&5
- z = 7&8

STANDARD

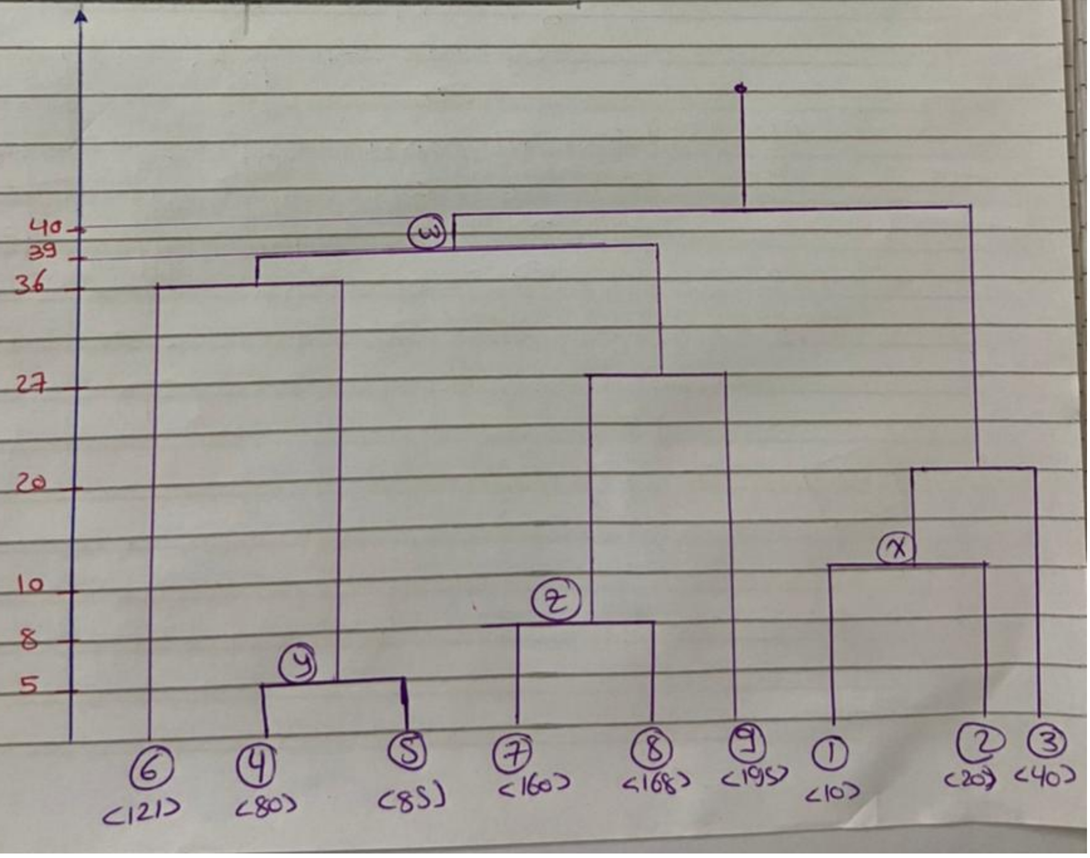
	x&3	y	6	z&9
x&3	0			
y	40	0		
6	81	36	0	
z&9	120	75	39	0

	x&3	y&6	z&9
x&3	0		
y&6	40	0	
z&9	120	39	0

let

• ω = y&6 + z&9

	x&3	ω
x&3	0	
ω	40.	0

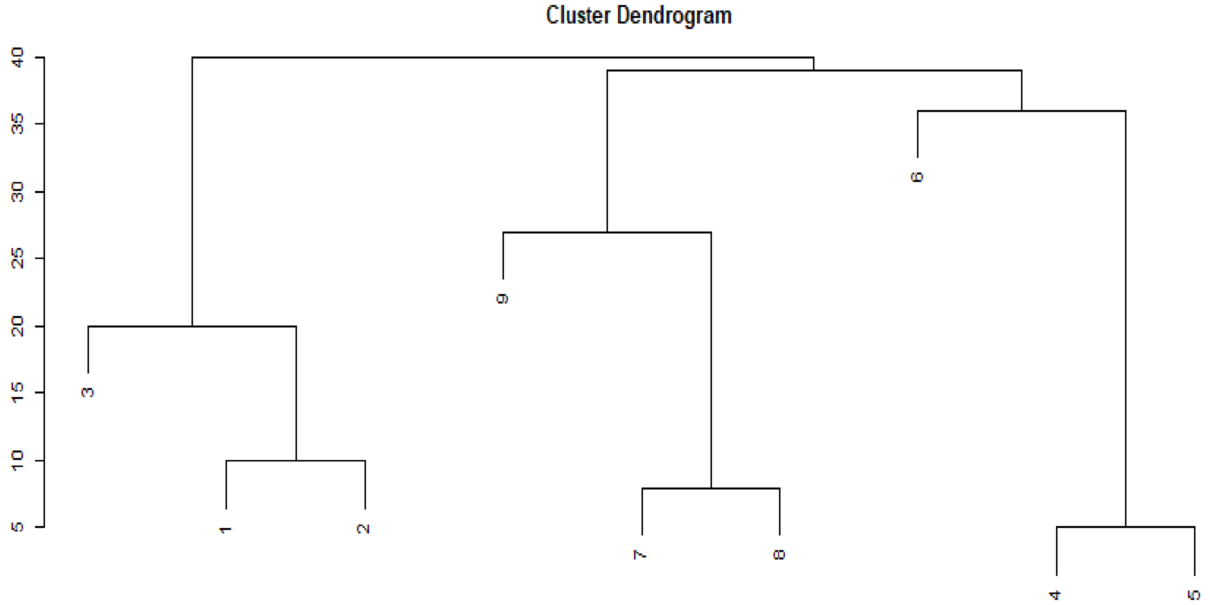


Code Results →

Single

Heights → [1] 5 8 10 20 27 36 39 40

Dendogram →



Complete

STANDARD

2) Complete linkage : => Start by the same way; take 8 → 4&5 as one cluster

	1	2	3	4&5	6	7	8	9
1	0							
2	10	0						
3	30	20	0					
4&5	70	65	45	0				
6	111	101	81	41	0			
7	150	140	120	80	39	0		
8	158	148	128	88	47	8	0	
9	185	175	155	115	74	35	27	0

	1	2	3	4&5	6	7&8	9
1	0						
2	10	0					
3	30	20	0				
4&5	70	65	45	0			
6	111	101	81	41	0		
7&8	158	148	128	88	47	0	
9	185	175	155	115	74	35	0

	1&2	3	4&5	6	7&8	9
1&2	0					
3	30	0				
4&5	70	45	0			
6	111	81	41	0		
7&8	158	128	88	47	0	
9	185	155	115	74	35	0

let $x = 1,2$
$y = 4,5$
$z = 7,8$

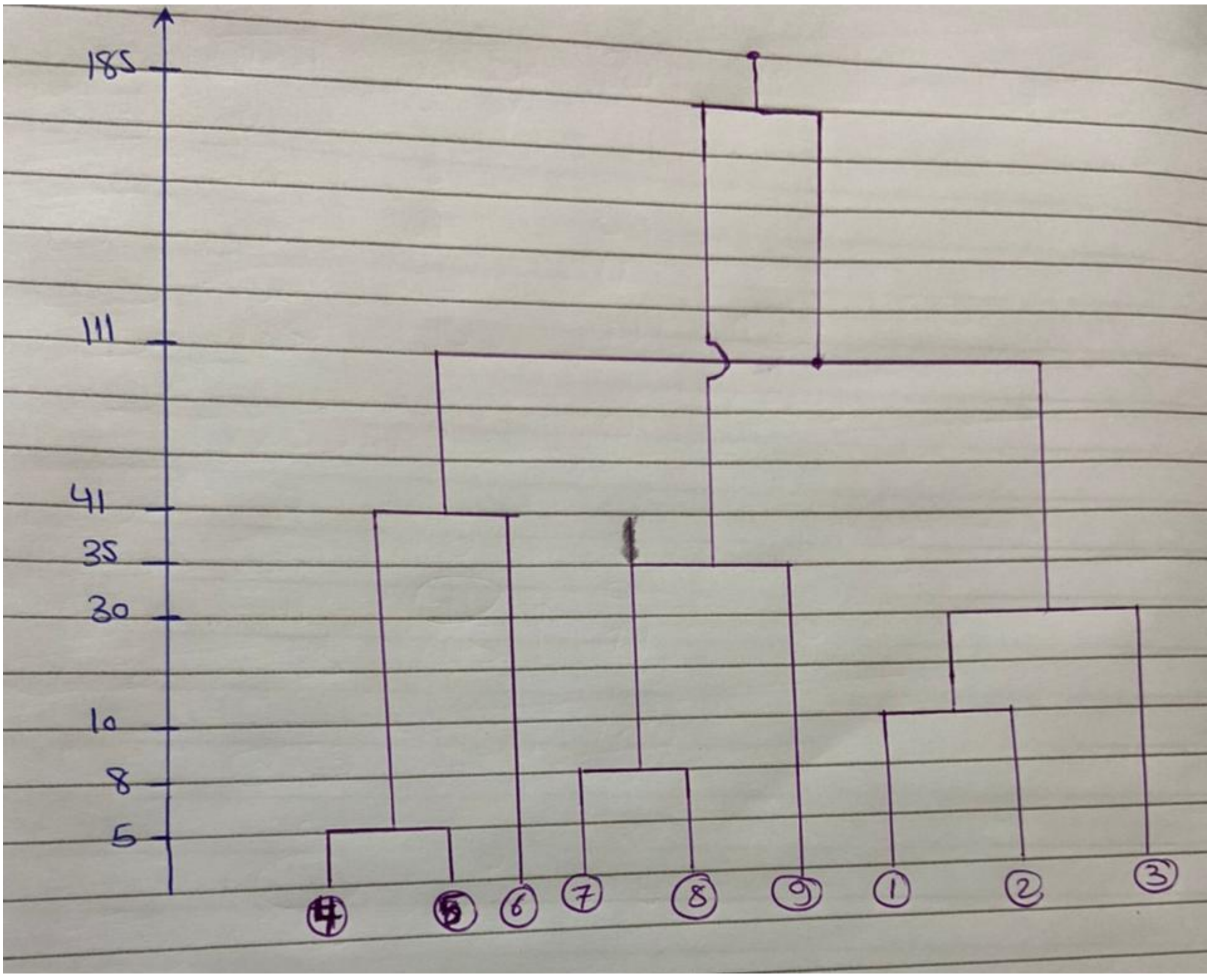
	X&3	4&5	6	7&8	9
X&3	0				
4&5	70	0			
6	111	41	0		
7&8	158	88	47	0	
9	185	115	74	35	0

	X&3	4&5	6	Z&9
X&3	0			
4&5	70	0		
6	111	41	0	
Z&9	185	115	74	0

	X&3	Y&6	Z&9
X&3	0		
Y&6	111	0	
Z&9	185	115	0

· let X&3 + Y&6 = W

	W	Z&9
W	0	
Z&9	185	0



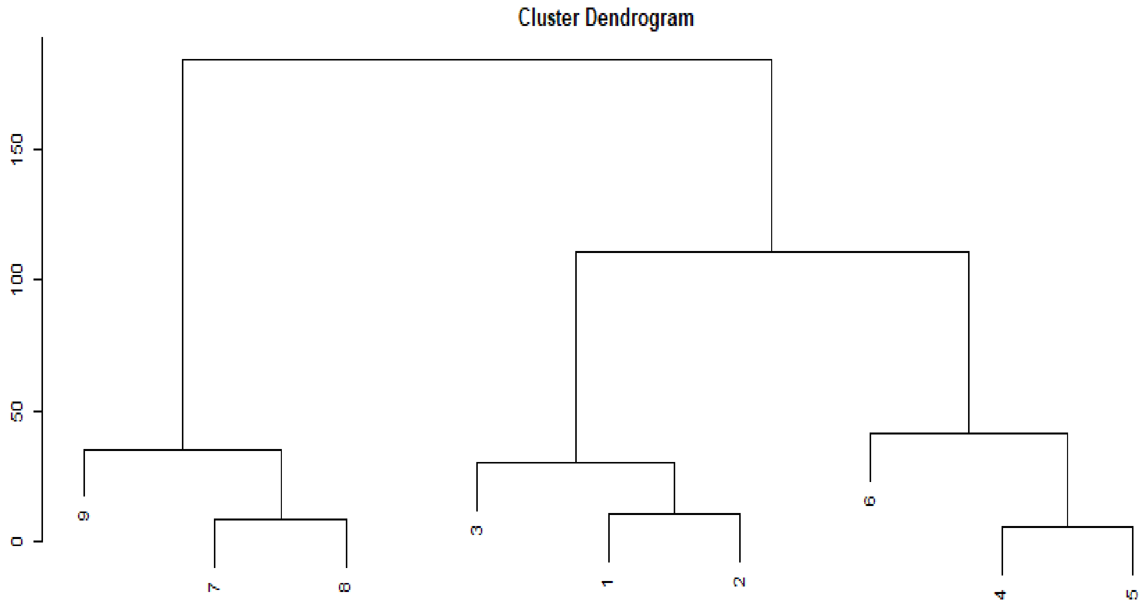
185
111
41
35
30
10
8
5
4
5
6
7
8
9
1
2
3

Code Results:

Complete

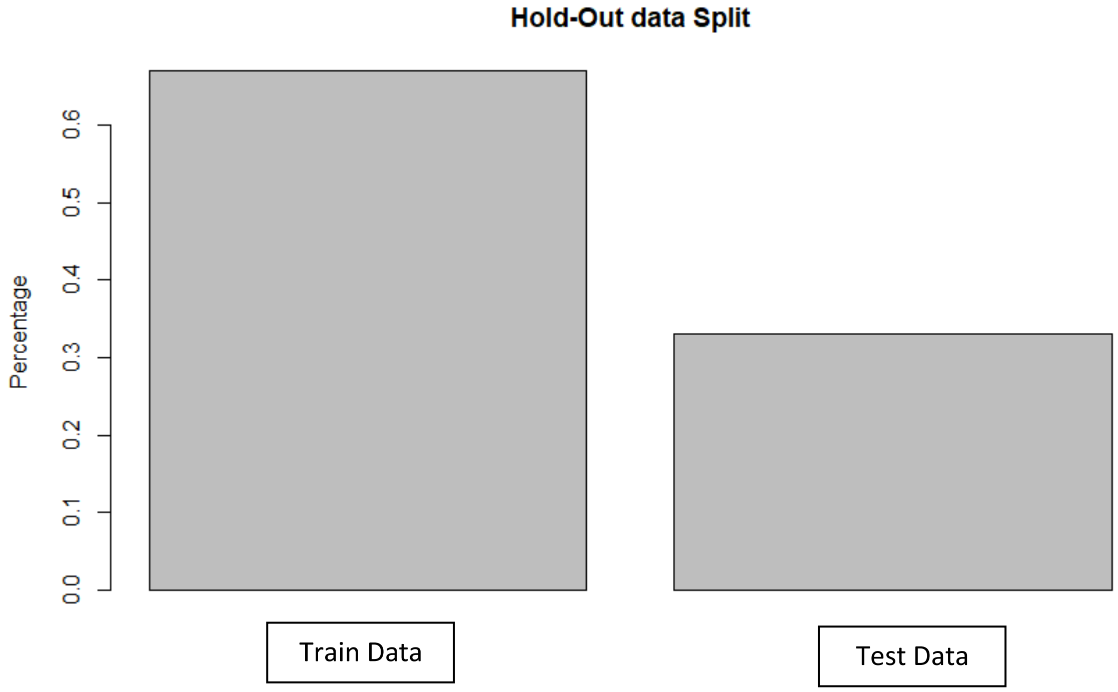
Heights → [1]   5   8   10   30   35   41 111 185

Dendogram →

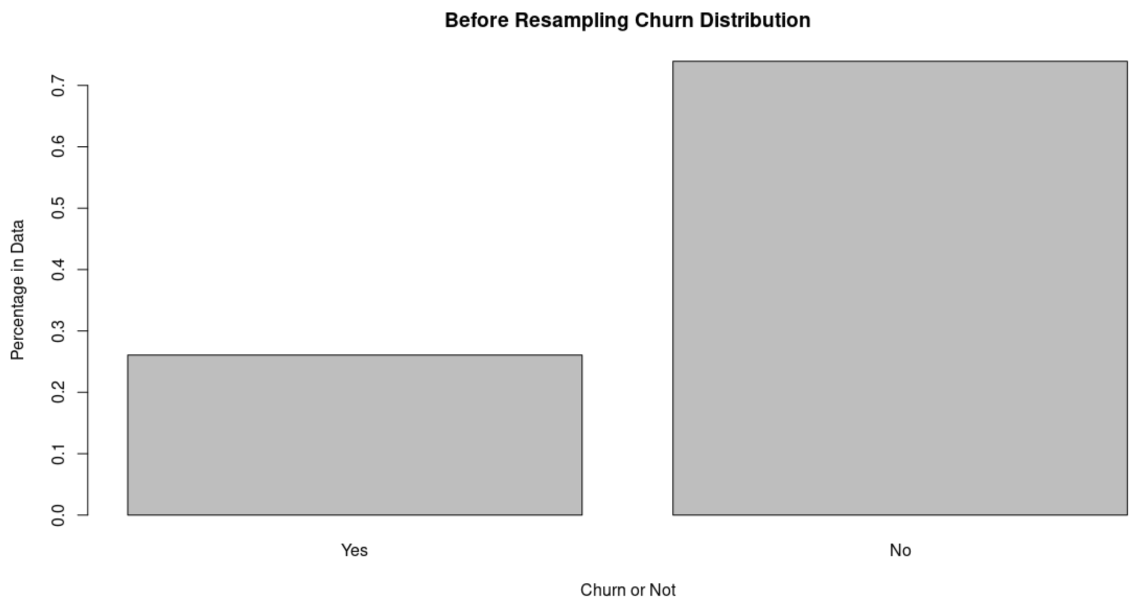


Part B: Evaluation & Improvement

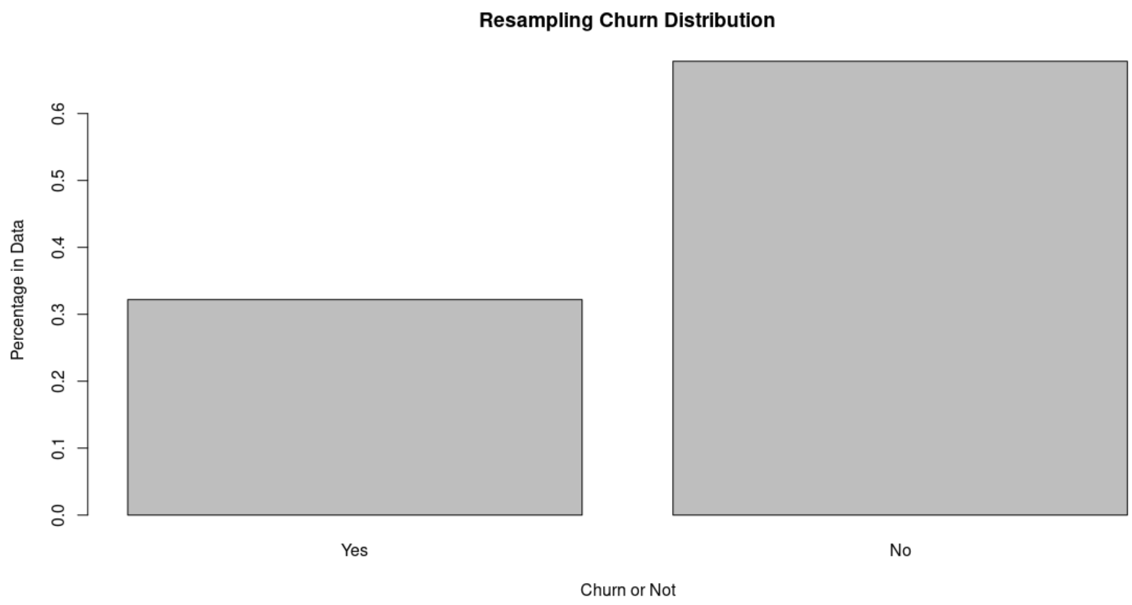
a) Partitioning the data into train_set (67%) and test_set (33%)



b) Total Number of records → 4718
True Churn (1) → 26% (1226)
Distribution of Churn values before resampling →



c) Resampling and confirming the distribution of Churn values afterwards as we oversample the minority ‘Yes’ Class to be 30% of data. (ROSE library was used)



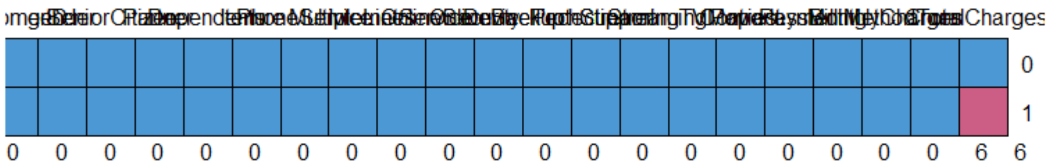
d) The predictors appropriate for decision tree:

1- Customer ID → this feature doesn’t seem to be significant for this problem so we drop it.

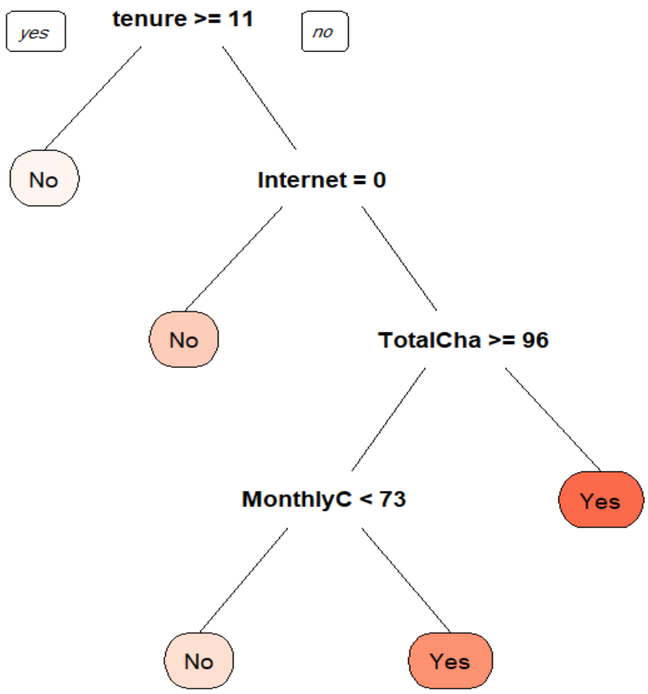
2- Using the varImp to check the importance of each feature influencing the target: we can see that some features aren’t important as gender and Dependents so we drop them (all that have importance of zero)

```
> print(importance)
                    Overall
Contract         180.138039
InternetService  173.910261
MonthlyCharges    62.267128
MultipleLines      6.050557
OnlineBackup       6.771847
OnlineSecurity   193.170981
PaymentMethod     35.077907
StreamingMovies   13.054249
StreamingTV        3.772471
TechSupport      180.819460
tenure           160.815320
TotalCharges      42.014791
gender             0.000000
SeniorCitizen      0.000000
Partner            0.000000
Dependents         0.000000
PhoneService       0.000000
DeviceProtection   0.000000
PaperlessBilling   0.000000
```

Note that: There were some missing values in ‘Total Charges’, about 6 rows, thus they were dropped.



Decision Tree → We can see that highly important features as tenure and Internet services were used in the Tree.



The predicted value on the test set along with its confusion matrix were obtained (will be shown in point f)

e) Here, I used ‘Random Forest’ as the Ensemble Method.
Some parameters were tuned → Number of trees = 100, each Node size = 5 and Maximum nodes = 10
The accuracy increased immensely after the tuning along with other metrics.
Accuracy before tuning = 0.77
Accuracy after tuning = 1

f) We have 3 models here: Decision Tree, RandomForest -before tuning-, RandomForest -after tuning- (Here 1/positive→ No,   2/Negative → Yes)

Decision Tree:

```
Confusion Matrix and Statistics

          Reference
Prediction    1    2
         1 1671  466
         2   55  133

               Accuracy : 0.7759
                 95% CI : (0.7584, 0.7927)
    No Information Rate : 0.7424
    P-Value [Acc > NIR] : 9.697e-05

                  Kappa : 0.2451

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9681
            Specificity : 0.2220
         Pos Pred Value : 0.7819
         Neg Pred Value : 0.7074
             Prevalence : 0.7424
         Detection Rate : 0.7187
   Detection Prevalence : 0.9191
      Balanced Accuracy : 0.5951

       'Positive' Class : 1
```

Random Forest (**Before** Tuning)

```
          Reference
Prediction    1    2
         1 1750  524
         2    7   39

               Accuracy : 0.7711
                 95% CI : (0.7535, 0.7881)
    No Information Rate : 0.7573
    P-Value [Acc > NIR] : 0.06286

                  Kappa : 0.0949

 Mcnemar's Test P-Value : < 2e-16

            Sensitivity : 0.99602
            Specificity : 0.06927
         Pos Pred Value : 0.76957
         Neg Pred Value : 0.84783
             Prevalence : 0.75733
         Detection Rate : 0.75431
   Detection Prevalence : 0.98017
      Balanced Accuracy : 0.53264

       'Positive' Class : 1
```

Here, even though the sensitivity (the **truly** predicted as **positive** cases) increased, however the specificity decreased (the **truly** predicted as **negative** cases) Which is plausible to be the issue as the specificity is the metric that measures the 'Yes' in our data which was already less represented -even after resampling- that's why the overall accuracy didn't change much but slightly decreased for the dramatically lower specificity.

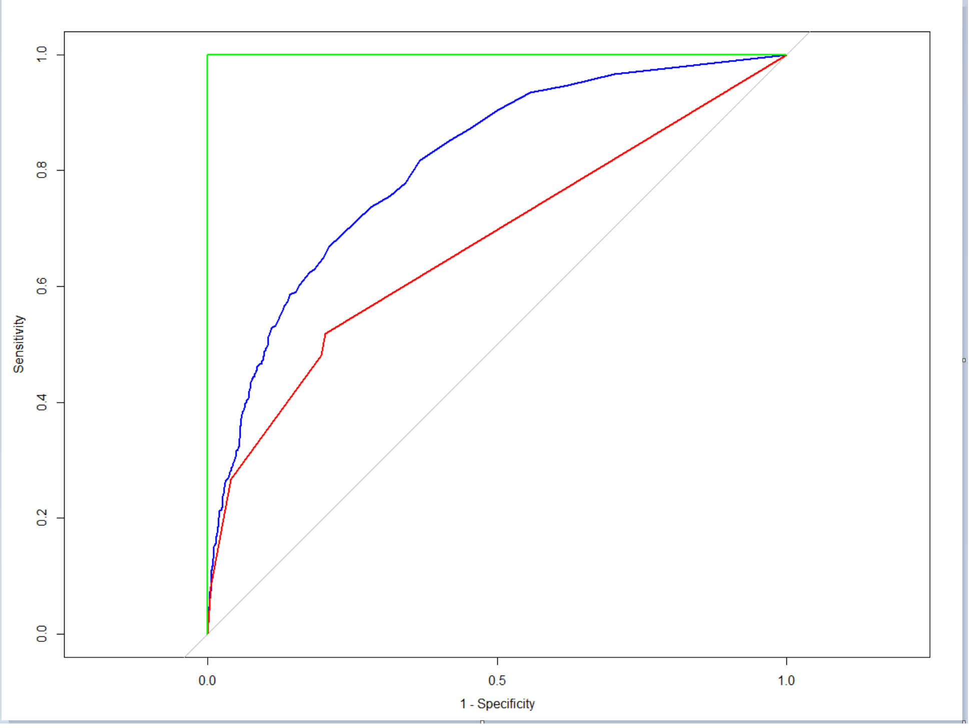
Random Forest (**After** tuning)

```
          Reference
Prediction    1    2
         1 1724    0
         2    0  599

               Accuracy : 1
                 95% CI : (0.9984, 1)
    No Information Rate : 0.7421
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 1

 Mcnemar's Test P-Value : NA

            Sensitivity : 1.0000
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 1.0000
             Prevalence : 0.7421
         Detection Rate : 0.7421
   Detection Prevalence : 0.7421
      Balanced Accuracy : 1.0000

       'Positive' Class : 1
```

Here the model is perfect, it managed to predict everything correctly, so the accuracy, sensitivity and specificity are optimal = 1

Metric	**Decision Tree**	**RF (no Tune)**	**RF (Tuning)**
Accuracy	0.776	0.771	1
Sensitivity	0.96	0.99	1
Specificity	0.22	0.069	1
	Moderate in all	Worst Specificity	Best in all

g) ROC curves for the 3 models:
Red → Tree,
Blue → Random Forest without tuning,
Green → Random Forest with tuning



Obviously, here the ROC is perfect for RF with tuning, where the AUC is maximum.