MATH501 Modelling and Analytics for Data Science Coursework

10546918

28 April 2021

1 Machine Learning Task

This report aims to construct a classification prediction model, in order to predict if a customer is going to stay or switch to a different telecommunication provider.

1.1 Machine Learning Part (a)

1.1.1 Data exploration

The data set has information about 500 costumers of a telecom company. It consists of 500 rows and 5 columns.

Table 1: Head of the data set

upload	webget	enqcount	callwait	churn
9.2	283.9	5	8.14	no
7.0	298.4	6	11.59	no
6.6	163.8	5	8.25	no
15.0	566.8	2	9.50	no
11.1	210.3	5	6.96	no
15.4	857.0	2	10.80	yes

The tables below gives us statistical information regarding the variables of our dataset.

Table 2: Mean values of the variables (group by churn)

churn	mean_upload	mean_webget	mean_callwait
no	10.15113	331.6796	8.090957
yes	12.71456	535.4456	9.087282

Table 3: Summary Statistics

upload	webget	enqcount	callwait	churn
Min. : 4.30	Min. : 136.4	Min. :0.000	Min. : 0.000	no :397
1st Qu.: 8.50	1st Qu.: 248.4	1st Qu.:2.000	1st Qu.: 6.900	yes:103
Median :10.80	Median : 326.6	Median :3.000	Median : 8.460	NA
Mean :10.68	Mean : 373.7	Mean :3.386	Mean : 8.296	NA

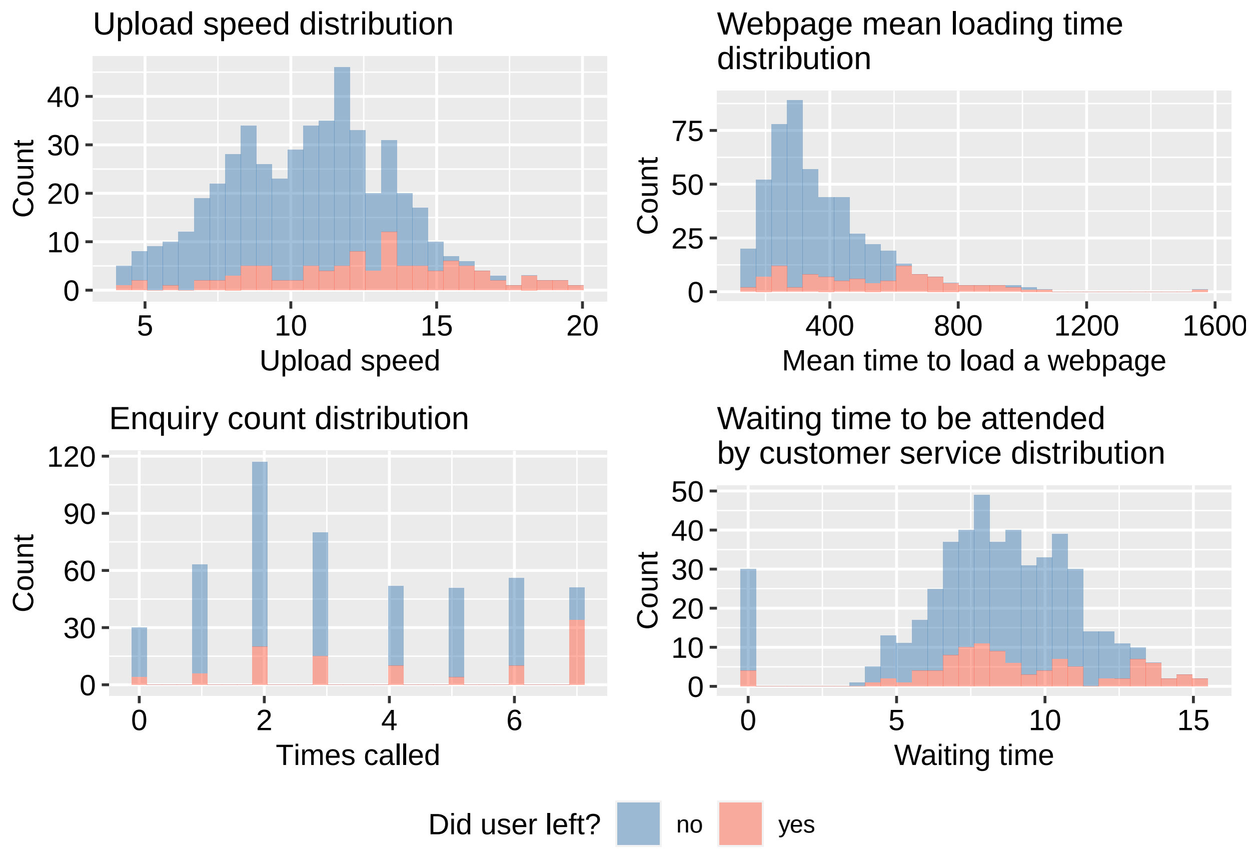
upload	webget	enqcount	callwait	churn
3rd Qu.:12.60	3rd Qu.: 450.9	3rd Qu.:5.000	3rd Qu.:10.330	NA
Max. :19.80	Max. :1544.4	Max. :7.000	Max. :15.230	NA

From **table 2** above we can notice that there is a difference in the means of the customers who switched to another telecommunication provider.

Furthermore it is important to say, that we have an unbalanced data set, since the customers who switched to a different provider account for only the 20.6% of our sample. An unbalanced data set can affect our model's performance. Models trained on unbalanced datasets usually suffer from poor accuracy when have to generalize, since the algorithm has fewer examples of the minority class, thus will be more biased towards the majority class.

1.1.2 Data Visualization

Histograms

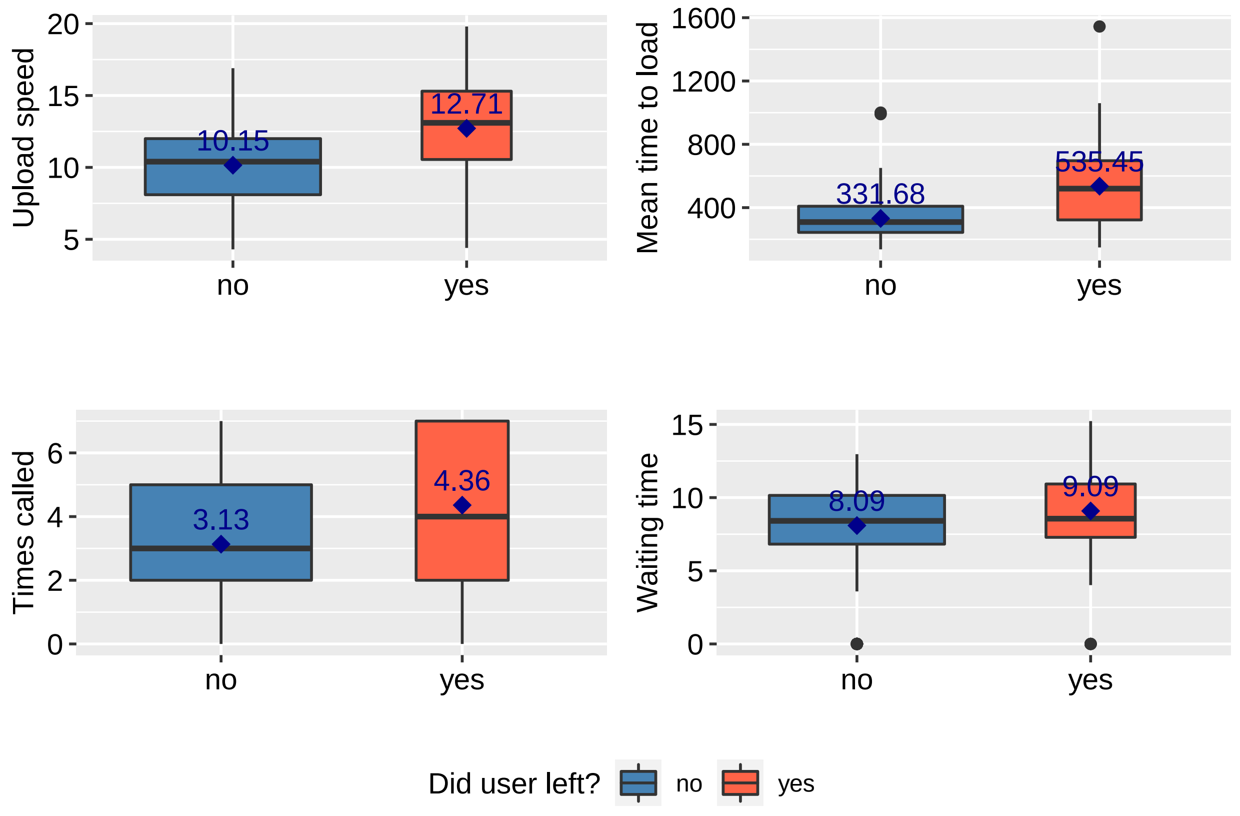


While observing the histogramms we can conclude the following:

1. The majority of the customers, have not switched to another provider.
2. The proportion of customers who have switched, tend to have higher upload speed.
3. The mean time to load a webpage, for the majority of the customers who remained loyal to the company, is between 200 - 400. If the value increases, the chance for a customer to switch provider increases.
4. The customers who have called the company 7 times tend to switch provider.

5. The distribution of waiting time for the both classes is similar, however for values above 12.5 there are more customers who switched provider.

Boxplots



From the boxplots we notice the below:

1. The boxplot of uploading speed for those who have not switched, is relatively narrow, indicating that overall there is a small degree of variation of their upload speeds. Those who have switched provider, have a higher variance of upload speed. Furthermore it is **clear that those who have switched tend to have higher upload speeds**, while their mean value is about 2.5 points higher, and the max observed value is about 3 points greater from the max speed of the other group. Also the customers falling within the IQR for the 'no' class, indicates that 50% of them have speed between about 7.5 and 12.5, while for the 'yes' class from about 10 (which is almost the median of the 'no' class) to above 15. For both classes upload speeds below 5 have been reported.

2. We notice that the boxplot indicating the mean time to load a webpage is too narrow (whiskers included), for the group who stayed in the company, indicating a very small variance. In more details the 75% of this group witness mean waiting time about 180-400. There are few outlines at the value 1000, for both classes. On the other hand we notice, that the loading time varies more for those who switched providers, since 50% of them experience loading times from about 380 to 700. The upper quartile of the 'no' class ends only a little higher where the lower quartile of the 'yes' class starts. We can clearly state that **the group who switched provider have greater waiting times**, as the greater median, mean and previous mentioned analysis indicates.

3. Those who have not switched provider, have contracted less times the company in average (3.13), compare to those who have (4.36). This can be explained, because customers facing connection issues

tend to call more, and if their issues are not being solved they tend to switch providers. Also we notice that the max times someone have called the company is 7 times. Furthermore the boxplot for the group who switched provider have no upper whisker, since the limit of the IQR is the same with the max observed value. This can be explained by the extreme skewness of the data (the majority of the values of this class are 7). Finally we notice that there are some customers in both classes who have never called, while there are significant less from 'no' class who have called more than 5 times compared to the 'yes'. The IQR range of the 'no' class is between 2 - 5 calls, while for the 'yes' from 2-7. In general **we can argue that those who switched provider tend to call more times**.

4. There appears not to be any great difference of the waiting time between the two groups, while the median seems the same (about 8). The IQR of those who switched provider, is little longer (from 7.5 to 11), and the whiskers as well, indicating higher variation. The max value for those who switched is about 15, while for the other group 12.5. Both groups show some outlines for the value of 0, indicating that many customers did not waited (maybe because they never called) to be attended by a customer service operator. In general **callwait appears not to be a significant factor**, however there is a little evidence that those who switched provider tend to have experienced longer waiting times.

1.2 Machine Learning Part (b)

In order to produce a classification model, we firstly randomly split our data set into a training set, containing 350 randomly chosen data points, and a test set containing the remaining 150 data points.

1.3 Machine Learning Part (c)

Firstly we are going to use the $k - NearestNeighbors(KNN)$ classifier to make predictions.

The KNN classifier makes predictions based on the distance (Euclidean distance in our case), thus the scale of the variables matters. If we do not scale the data, variables with high variance (wider range), will effect more the Euclidean distance and will appear to be more important for the determination the target value. Since we do not know the scales of our variables, its wise to scale our data. However, since our variables always get positive numbers as values, we used the max-min normalization. We have to keep in mind normalization does not handle the outlines well, but according to our previous analysis there are not many.

We will use the formula below to normalize each value of our variables (except of the target variable 'churn').

$$x' = \frac{x - min(x)}{max(x) - min(x)}$$

Where:

- x': the normalized value
- x: the original value

Firstly we are going to use the plain **knn()** function to make predictions, and then compare it with the leave-one-out cross-validation method.

```
library(class)
knn_1 <- knn(train = df_train_tele_norm[,-5],
                      test = df_test_tele_norm[,-5],
                      cl=df_train_tele_norm$churn,
                      k=1)
knn_1_accuracy <- mean(knn_1 == df_test_tele_norm$churn)
knn_1_error <- mean(knn_1 != df_test_tele_norm$churn)
```

```r
knn_1_conf_matrix <- table(knn_1, df_test_tele_norm$churn)

rownames(knn_1_conf_matrix) <- c("No","Yes")
kable(knn_1_conf_matrix,
       col.names = c('No','Yes'),
      caption = 'Confusion Matrix for k=1')
```

Table 4: Confusion Matrix for k=1

	No	Yes
No	114	31
Yes	2	3

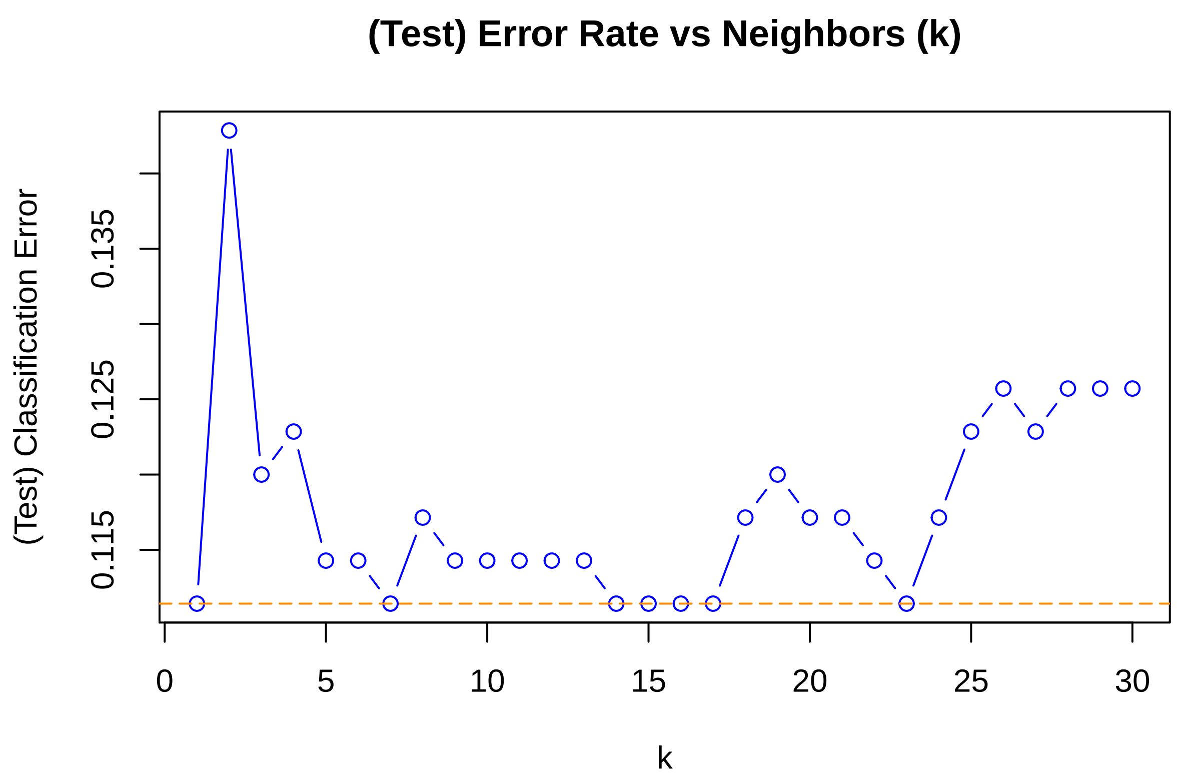
The test error for the KNN algorithm for $k = 1$ is 22%. We notice that the algorithm struggles to predict correctly those who switched provider (true positives), while the majority of the values are assigned as no, resulting 31 false negatives, thus a recall of of only 8.82%. This issue can be explained by the imbalance of our data set.

1.3.1 Leave-One-Out Cross-Validation

In order to find the optimal value of k, we are going to use the leave-one-out cross-validation for the training data set.

According to the graph below, for $k = 1$ or $k = 7$ we have the same error, however it is important to check the confusion matrices.

```r
set.seed(1)
cross_knn_error <- c()
error <- c()
for(i in 1:100){
  cross_knn <- knn.cv(df_train_tele_norm[,-5],
                      df_train_tele_norm[,5],
                      k = i,
                      l = 0,
                      prob = FALSE,
                      use.all = TRUE)
  cross_knn_error[i] <- mean(cross_knn != df_train_tele_norm[,5])
}
```



```r
set.seed(1)
cross_knn_7 <- knn.cv(df_train_tele_norm[,-5],
                      df_train_tele_norm[,5],
                      k = 7,
                      l = 0,
                      prob = FALSE,
                      use.all = TRUE)
cross_knn_7_error <- mean(cross_knn_7 != df_train_tele_norm[,5])
cross_knn_7_accuracy <- mean(cross_knn_7 == df_train_tele_norm[,5])
cross_knn_7_conf_matrix <- table(cross_knn_7, df_train_tele_norm[,5])

cross_knn_1 <- knn.cv(df_train_tele_norm[,-5],
                      df_train_tele_norm[,5],
                      k = 1,
                      l = 0,
                      prob = FALSE,
                      use.all = TRUE)
cross_knn_1_error <- mean(cross_knn_1 != df_train_tele_norm[,5])
cross_knn_1_conf_matrix <- table(cross_knn_1, df_train_tele_norm[,5])

rownames(cross_knn_7_conf_matrix) <- c("No","Yes")
kable(cross_knn_7_conf_matrix,
       col.names = c('No','Yes'),
      caption = 'Confusion Matrix for k=7')
```

Table 5: Confusion Matrix for k=7

	No	Yes
No	278	36
Yes	3	33

```r
rownames(cross_knn_1_conf_matrix) <- c("No","Yes")
kable(cross_knn_1_conf_matrix,
       col.names = c('No','Yes'),
      caption = 'Confusion Matrix for k=1')
```

Table 6: Confusion Matrix for k=1

	No	Yes
No	269	27
Yes	12	42

By calculating using the LOOCV, for both values of k, the test error is 11.14%. We notice that it performs really better compared to the simple KNN case. However, despite the fact that for $k = 1$ the model predicts better the true positives (those who switched providers) resulting a recall 60.87% compared to the 47.83% for $k = 7$, fewer $k - neighbors$ correspond to higher model complexity (Müller & Guido, 2017), meaning that our model might not generalize. Thus we would prefer the value for $k = 7$.

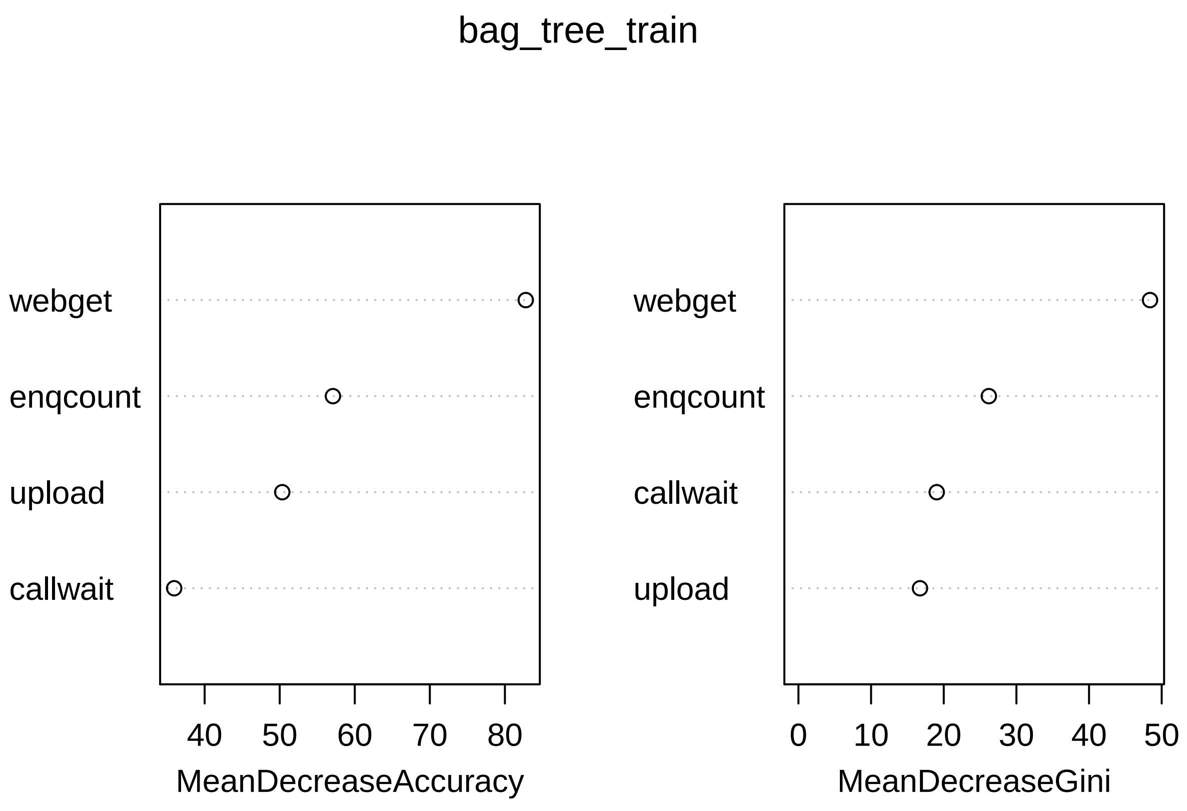
1.4 Machine Learning part (d)

We are going to implement the random forest (bagging) method to construct another classifier.

1.4.1 Measuring the importance of the variables

The figures below shows us the importance of the variables for predicting churn using the random forest classifier.

```r
library(randomForest)
set.seed(1)
bag_tree_train <-randomForest(formula =  df_train_tele[,5] ~.,
                              data=df_train_tele[,-5],
                              mtry=4,
                              importance=TRUE,
                              ntree=500)
varImpPlot(bag_tree_train)
```



```r
kable(importance(bag_tree_train),
      caption = "Variables' importance")
```

Table 7: Variables' importance

	no	yes	MeanDecreaseAccuracy	MeanDecreaseGini
upload	52.88654	26.38121	50.33764	16.72898
webget	71.52089	66.32982	82.77710	48.39307
enqcount	32.38104	52.43259	57.08985	26.20678
callwait	28.85396	30.34121	35.94219	19.03444

- **Mean decrease Accuracy** reflects how much accuracy is lost by excluding the respective variable, thus the more the accuracy decrease the more important this variable is.
- **Mean decrease Gini** shows the total decrease in node impurity resulted by the splits of a given variable, averaged over all trees. If a variable is useful. it tends to split a node into pure single class nodes. Thus, removing such variable, results a decrease in the average purity gained due to this variable.

We notice that **webget** and **enqcount** results the most mean decrease accuracy and gini. Also, **upload** scores higher in the mean decrease accuracy than **callwait**, while **callwait** scores better in gini, but with smaller spread.
In overall, we could argue that **webget** and **enqcount** are the most significant variables for classifying with random forest if a customer will switch provider.

1.4.2 Making predictions

```r
bag_tree_pred <- predict(bag_tree_train, newdata = df_test_tele[,-5])
bag_tree_train_acc <- mean(bag_tree_pred == df_test_tele[,5])
bag_tree_train_error <- mean(bag_tree_pred != df_test_tele[,5])
rf_matrix <-table(bag_tree_pred, df_test_tele[,5])
kable(rf_matrix,
      caption = 'ConfMatrix for randomForest')
```

Table 8: ConfMatrix for randomForest

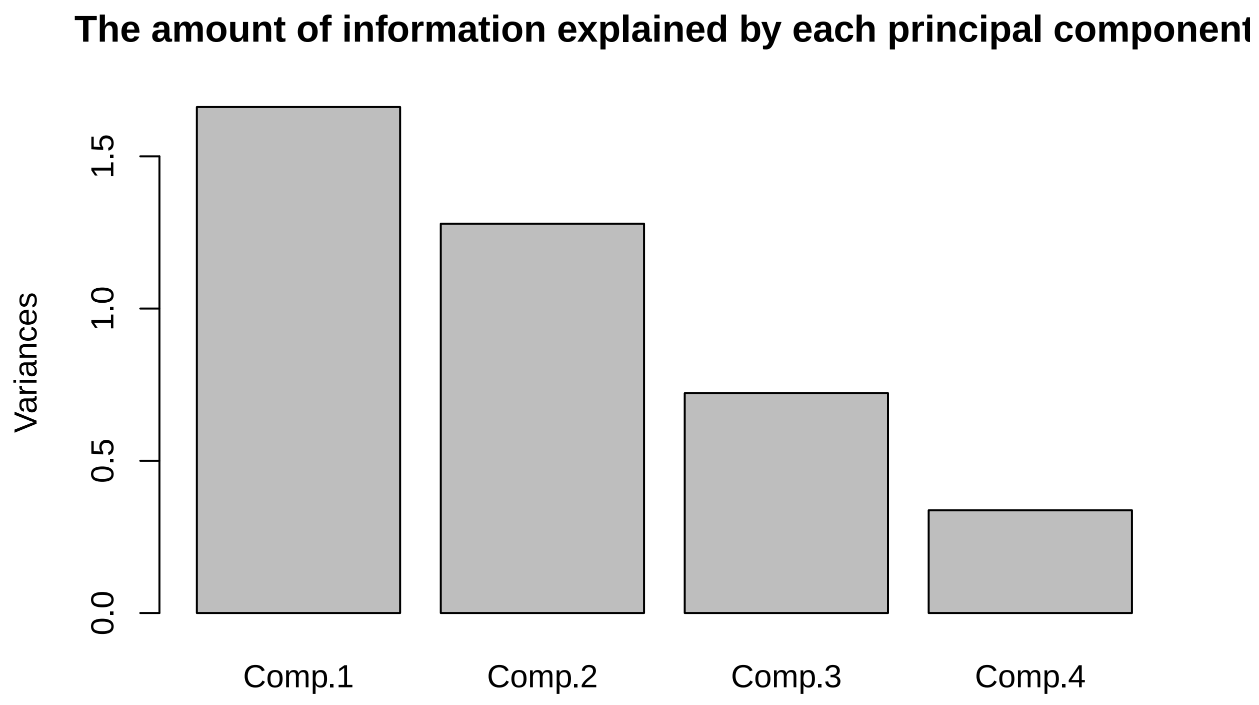
	no	yes
no	114	5
yes	2	29

We notice that the random forest performs better compared to the KNN models. The test error for the vanilla KNN with $k = 1$ is 22%, for the LOOCV KNN for $k = 7$ is 11.14% while for random forest the test error is only 4.67%, and thus its accuracy is 95.33%. We notice that the random forest performed much better predicting those who switched providers (True Positives), having a recall of 85.29%, compared to LOOCV KNN with $k = 7$ with recall = 47.83%. In general both KNN models, scored very poorly, on predicting those who switched providers. That might derives from the unbalance of our data set.

1.5 Machine Learning Part (e)

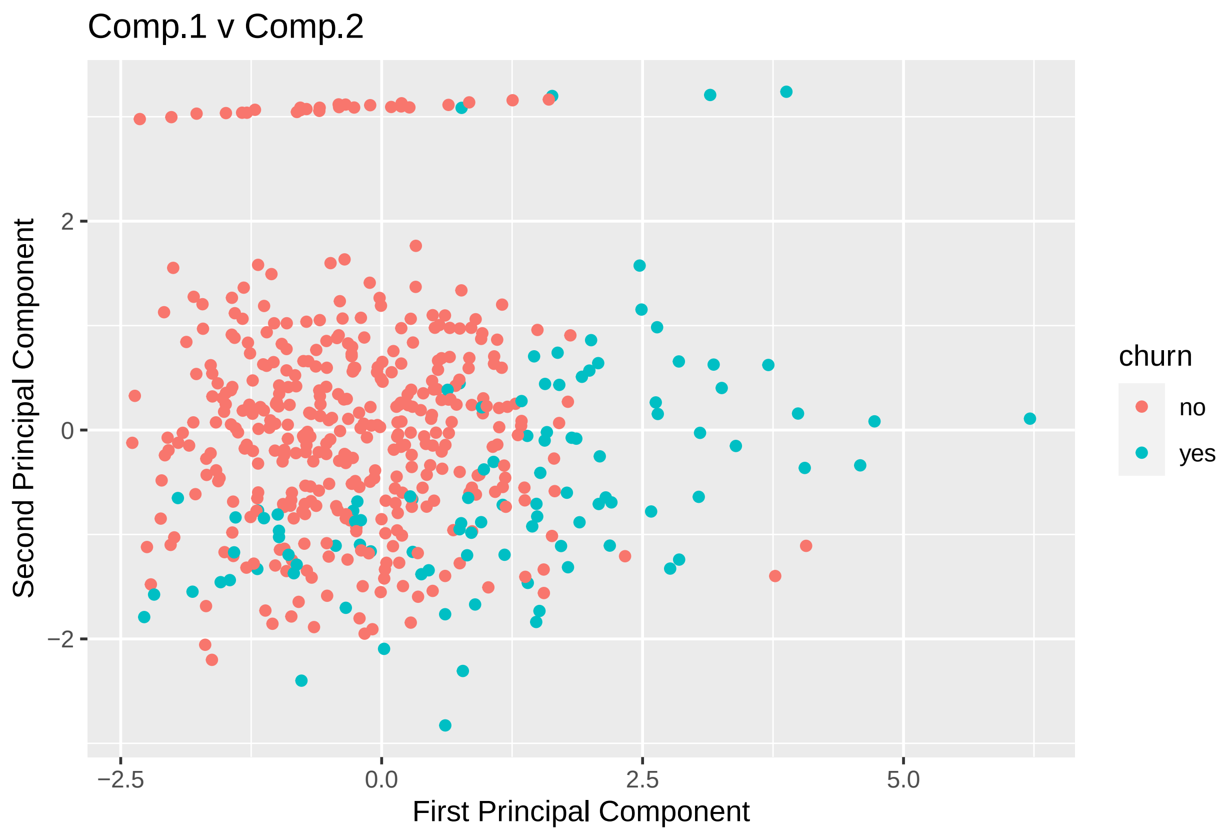
1.5.1 Principal Component Analysis (PCA)

We are going to perform Principal Component Analysis (PCA) for the four variables. PCA is a method usually used to reduce the dimensionality of a large data set of variables (Gareth et al., 2017).



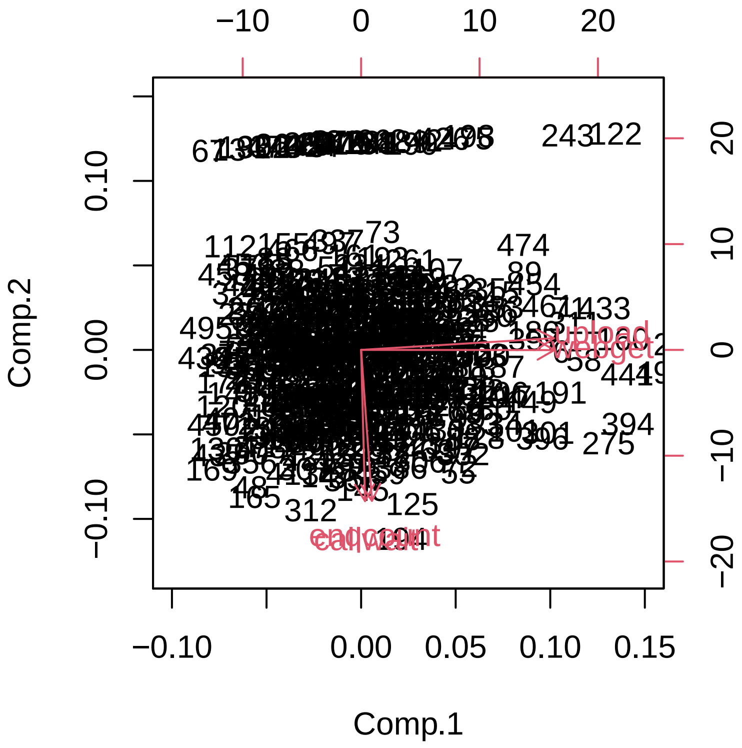
```
## Importance of components:
##                           Comp.1    Comp.2    Comp.3     Comp.4
## Standard deviation     1.2891552 1.1307782 0.8497143 0.58086584
## Proportion of Variance 0.4154803 0.3196648 0.1805036 0.08435128
## Cumulative Proportion  0.4154803 0.7351451 0.9156487 1.00000000
```

According to the summary table, Comp.1 accounts for the 41% of the information or variance in the data, Comp.2 for the 31%, Comp.3 for the 18% and Comp.4 for the 8%.

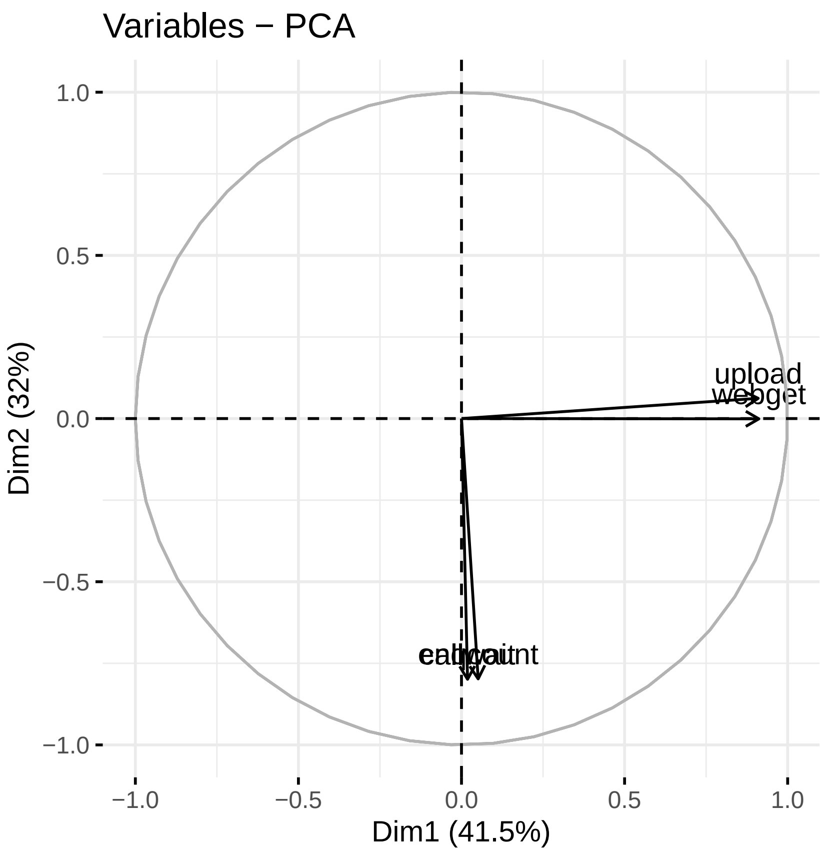


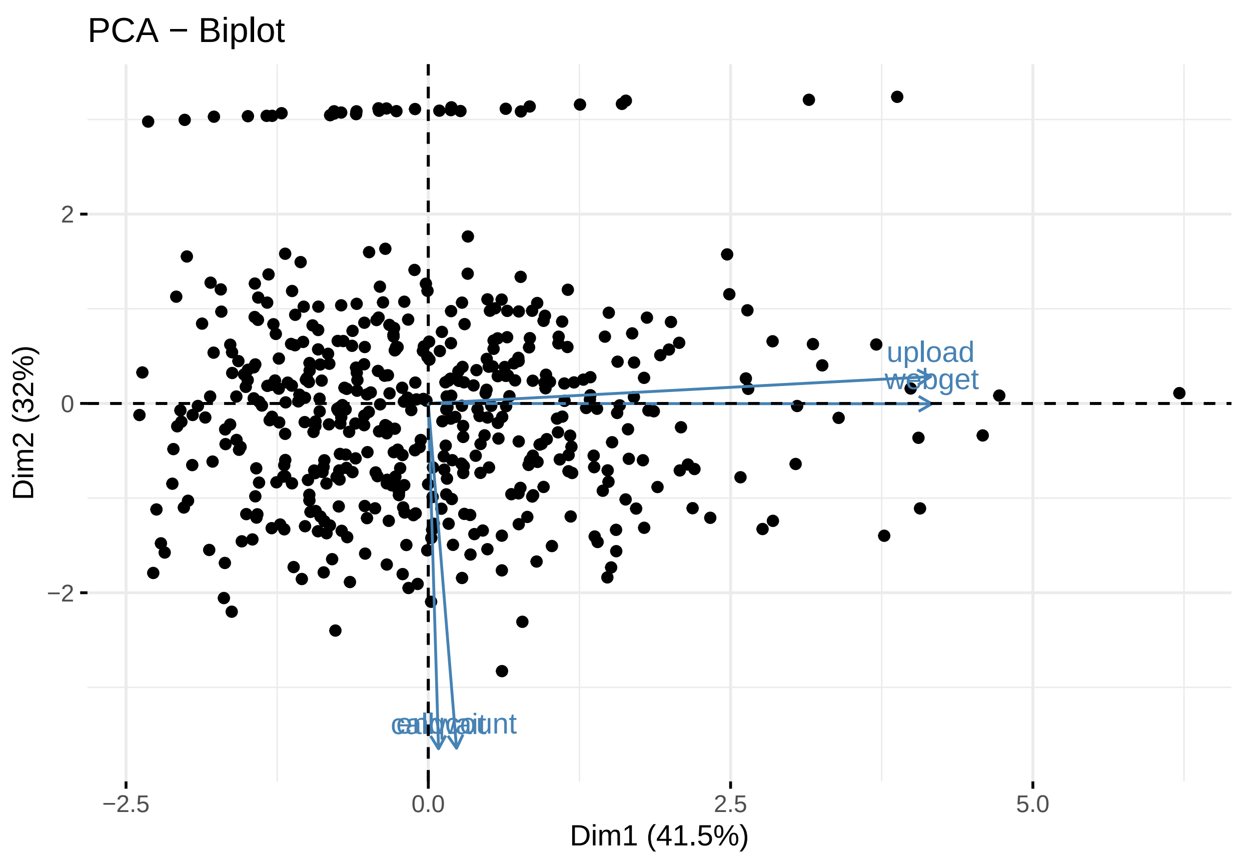
The information preserved in this plot equals to the proportion of the variances of the two components (Comp.1 and Comp.2) so is 73.5%

1.5.2 First two principal components interpretetion



We notice that the **biplot** function does not produce a clean plot, to this end for interpreting the first two principal components we implemented the **fviz_pca_biplot**, and **fviz_pca_var** functions from the **factorextra** package, since they produce a cleaner graph.


Variables − PCA
1.0
0.5
0.0
−0.5
−1.0
Dim2 (32%)
upload
webget
−1.0
−0.5
0.0
0.5
1.0
Dim1 (41.5%)



From the graphs above we notice that following:

1. The more parallel to a principal component axis is a vector, the more it contributes to that principal component. Thus, the variables **upload** and **webget** have very high (positive) contribution on the first principal component (the highest contribution is due to **webget**), while the variables **enqcount** and **callwait** almost do not contribute at all.
2. On the other hand, the variables **enqcount** and **callwait** have very high (negative) contribution on the second principal component, while the variables **upload** and **webget** almost do not contribute at all (**webget** appear to be inline with the x-axis so it will no contribute at all).
3. The longer the vector, the more variability of this variable is represented by the two displayed principal components. The variables **upload** and **webget** are longer than the **enqcount** and **callwait**, indicating that more variability of the former variables is presented by these two principal components, compared to the other two variables.
4. When two vectors form a small angle between them, it represents that the two variables are positively correlated. Thus, the **upload** and **webget** are positively correlated, as well as the **enqcount** and **callwait**. However as the graph indicates **upload** and **webget** with **enqcount** and **callwait** have no correlation between them, since their vectors form about a 90 degrees angle.

1.6 Machine Learning Part (f)

```
# splitting the data
comp_df_train <- new_variables[df_subset,]
comp_df_test <- new_variables[-df_subset,]
```

```
comp_forest <-randomForest(formula = churn ~.,
                           data=comp_df_train,
                           mtry=2,
                           importance=TRUE)
```

We are applying the random forest classifier (bagging) method to predict the value of **churn** variable based on the first two principal components.

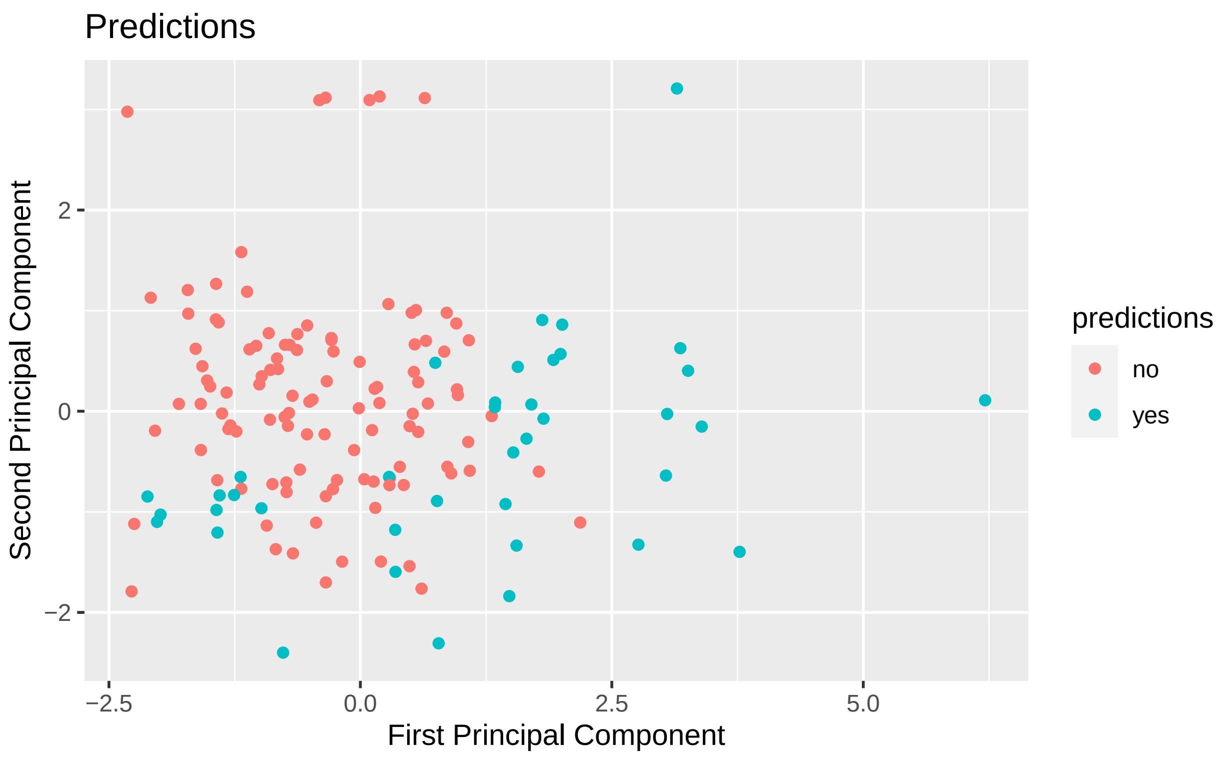
Table 9: Confusion matrix rf with components

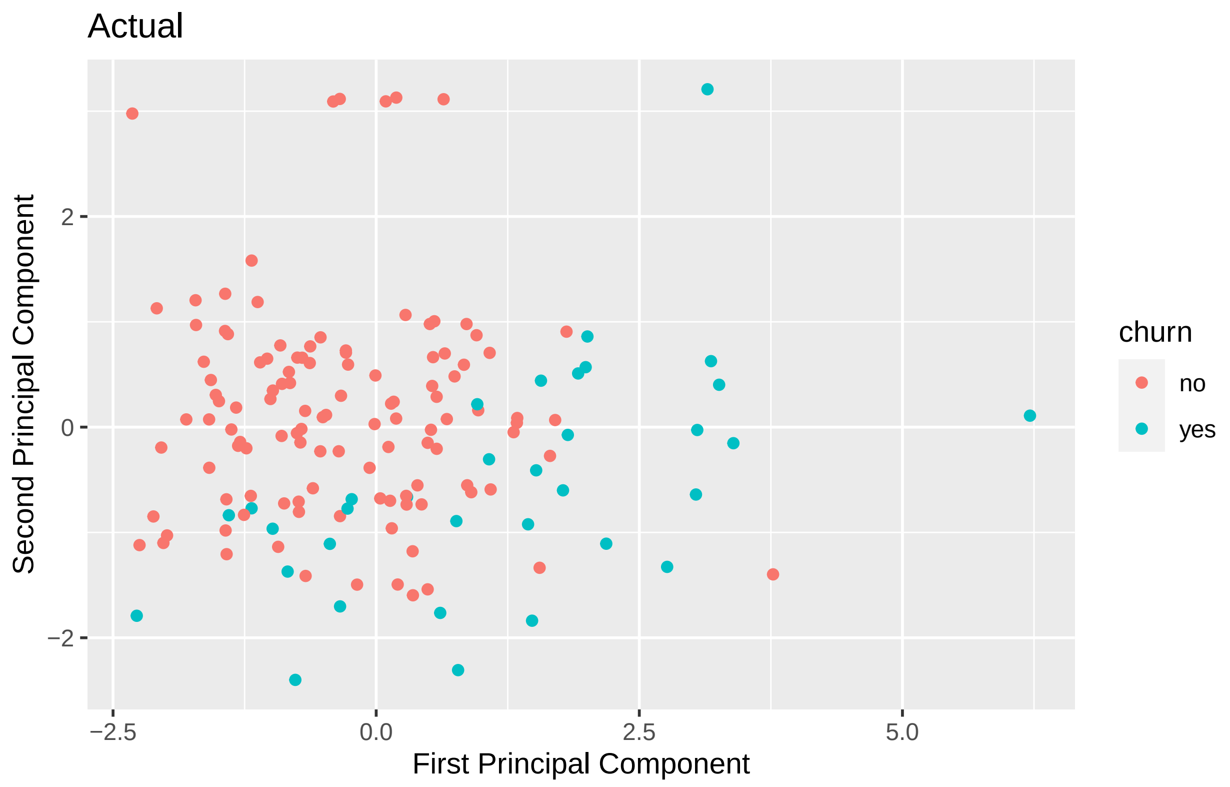
	no	yes
no	98	13
yes	18	21

The test error of random forest is 20.67%, which is worse than the test error obtained from the random forest which used all the variables (test error 4.67% part **d**).

In general we notice that, in this case the algorithm performs poorer in predicting the True Positives and True Negatives. However, if we take under consideration that the majority of the churn values are 'No', we notice that by this approach the model seems to be less affected by the majority class, since has assigned less values as 'No'. It has the lowest recall for the 'No' class compared to the rest of our models (84.48%).

The first graph below is an approach to show the resulting classification rule of this random forest, as a scatter plot of the two first principal components colored by the predicted class, while the second is colored by the actual values to show a comparison.





1.7 References

- James, G., Witten, D., Hastie, T. and Tibshirani, R., 2017. An introduction to statistical learning. 1st ed. New York: Springer.
- Muller, A.C. and Guido, S. 2016. Introduction to Machine Learning with Python. 1st ed. Sebastopol, CA: O'Reilly Media, Inc.