R Notebook

Your first membership inference attack

The first step is to load all the data. The data is available at: membership_inference_data.RData and place it in your working directory.

```
## load all required objects
load("C:/Users/Gilia/Dropbox/PhD/Conferences/2022/MARUG/workshop/membership_inference_data.rData")
```

The following files have been loaded into your working directory:

Churn data

```
print(head(churn2)) ## churn data set with 1,666 obs.
```

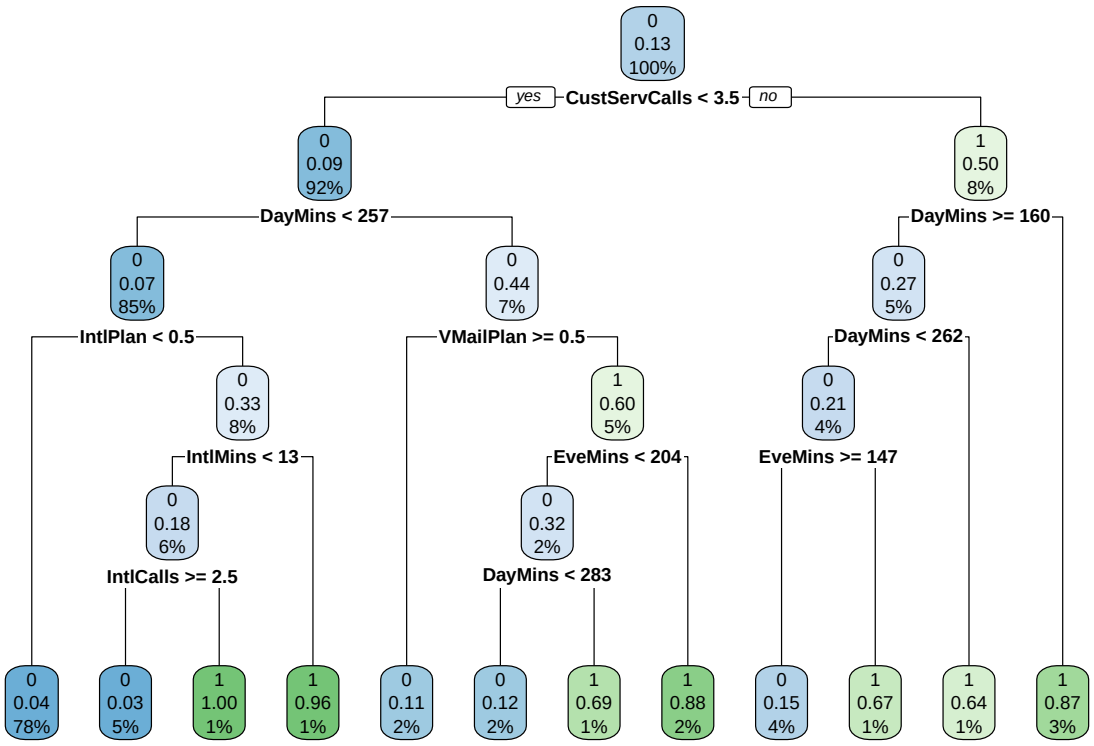
```
##   AccountLength IntlPlan VMailPlan VMailMessage DayMins DayCalls DayCharge
## 1           128        0         1           25   265.1      110     45.07
## 2           107        0         1           26   161.6      123     27.47
## 3           137        0         0            0   243.4      114     41.38
## 4            84        1         0            0   299.4       71     50.90
## 5            75        1         0            0   166.7      113     28.34
## 6           118        1         0            0   223.4       98     37.98
##   EveMins EveCalls EveCharge NightMins NightCalls NightCharge IntlMins
## 1   197.4       99     16.78     244.7         91       11.01     10.0
## 2   195.5      103     16.62     254.4        103       11.45     13.7
## 3   121.2      110     10.30     162.6        104        7.32     12.2
## 4    61.9       88      5.26     196.9         89        8.86      6.6
## 5   148.3      122     12.61     186.9        121        8.41     10.1
## 6   220.6      101     18.75     203.9        118        9.18      6.3
##   IntlCalls IntlCharge CustServCalls Churn
## 1         3       2.70             1     0
## 2         3       3.70             1     0
## 3         5       3.29             0     0
## 4         7       1.78             2     0
## 5         3       2.73             3     0
## 6         6       1.70             0     0
```

Trained model

Decision tree used to predict churn:

```
rpart.plot(tree)
```

```
## Warning: Cannot retrieve the data used to build the model (so cannot determine roundint and is.binary
## To silence this warning:
##     Call rpart.plot with roundint=FALSE,
##     or rebuild the rpart model with model=TRUE.
```



Use the trained model to predict churn over the entire data set.

The model was trained as follows:

$Churn = AccountLength + IntlPlan + VMailPlan + DayMins + DayCalls + EveMins + EveCalls + NightMins + NightCalls + IntlMins + IntlCalls + CustServCalls$

```
independent <- ("AccountLength + IntlPlan + VMailPlan + DayMins + DayCalls + EveMins +
EveCalls + NightMins + NightCalls + IntlMins + IntlCalls + CustServCalls")
BaseFormula <- as.formula(paste0("Churn ~ ", independent))
print(BaseFormula)
```

```
## Churn ~ AccountLength + IntlPlan + VMailPlan + DayMins + DayCalls +
##     EveMins + EveCalls + NightMins + NightCalls + IntlMins +
##     IntlCalls + CustServCalls
```

Next, we use the trained model to predict over all the observations.

```
predictions <- predict(tree, newdata = churn2, type = "prob") # obtain predictions
print("churn predictions")
```

```
## [1] "churn predictions"
```

```
head(predictions[1,2])
```

```
## [1] 0.1081081
```

```
print("true churn")
```

```
## [1] "true churn"
```

```
head(churn2$Churn[1])
```

```
## [1] 0
## Levels: 0 1
```

Calculating the error.

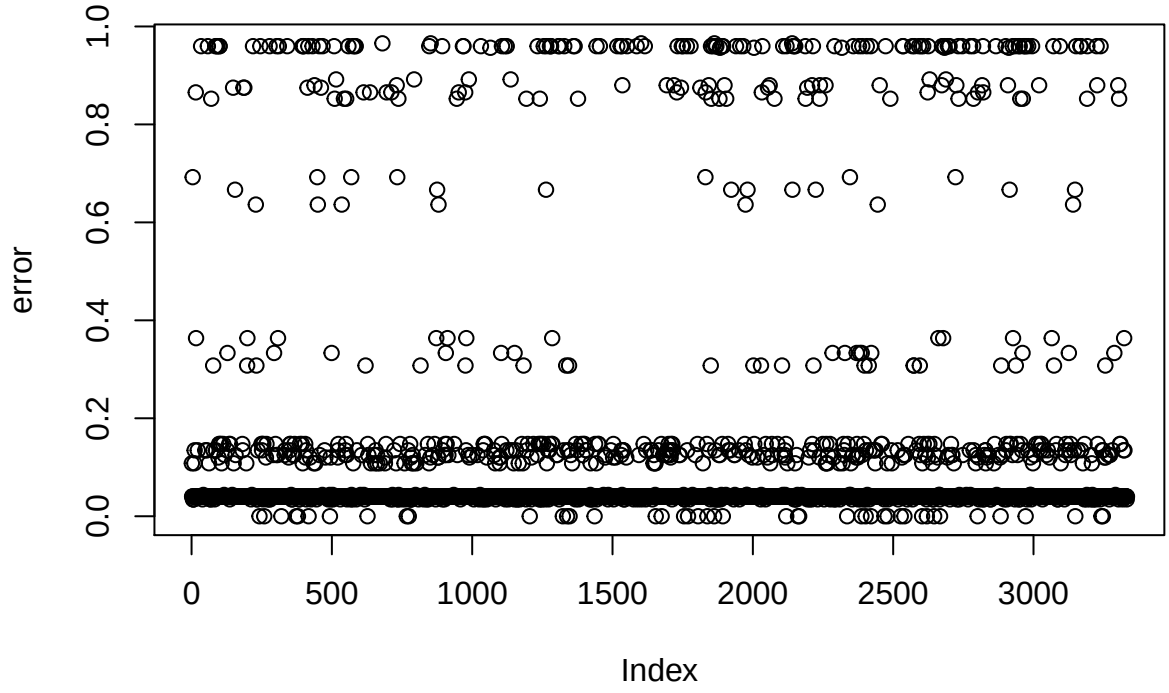
We use the predictions to calculate the error: $error = churn - predictions$.

```
churn2$Churn = as.numeric(churn2$Churn)-1
error = churn2$Churn - predictions[,2]
print(head(error))
```

```
##           1           2           3           4           5           6
## -0.10810811 -0.04024768 -0.04024768 -0.69230769 -0.03448276 -0.03448276
```

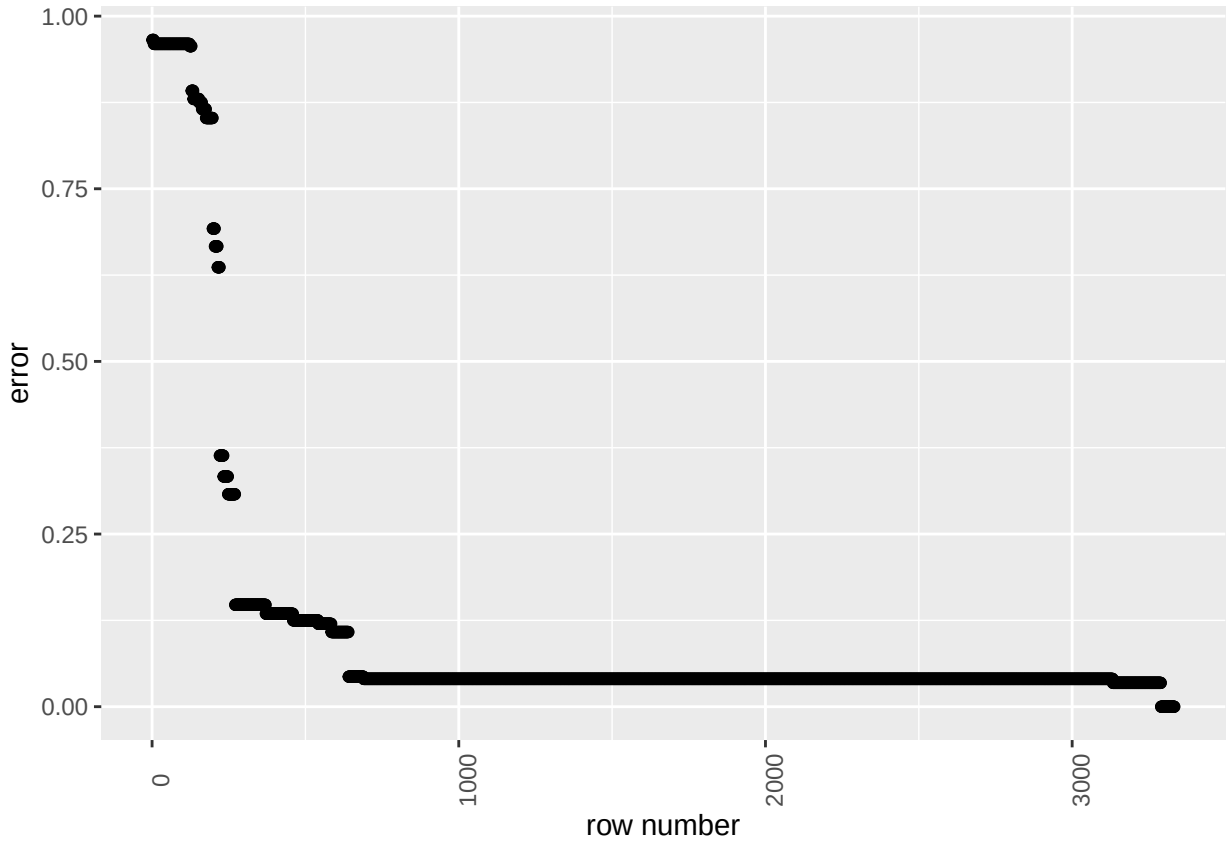
Take the absolute value of the error, sort the error from high to low.

```
error = abs(error)
plot(error)
```



We sort the error from low to high. Assuming that low error = training set!

```
sorted = data.frame(sort(error, decreasing =F)) # sort descending.
sorted = as.data.frame(setDT(sorted, keep.rownames = TRUE)[]) # row numbers to a column in data frame.
colnames(sorted) = c("rn","error")
ggplot(sorted, aes(x = as.numeric(reorder(rn,-error)), y = as.numeric(error))) + geom_point() + ylab("e
```



Simply select the first 1,666 observations and say they are in training set!

```
in_training = sorted[1:1666,] # get the first 1,666 observations that have the highest loss
in_training$rn = as.numeric(in_training$rn) # make row number numeric
in_training$train_prediction = 1 # assign label that the data point is in training set.
```

Combine the predictions with the original data set.

```
churn2$rn = as.numeric(row.names(churn2)) # store row number.
churn2$train = c(rep(1, 1666), rep(0,1667)) # assign true labels.
accuracy = left_join(churn2, in_training) # join predictions/error with the original data set, based on
```

```
## Joining, by = "rn"
```

```
accuracy[is.na(accuracy$train_prediction),22] = 0 # the values that are missing = 0. In other words, th
head(accuracy[,c(20,22)])
```

```
##   train train_prediction
## 1     1                0
## 2     1                1
## 3     1                1
```

```
## 4     1                0
## 5     1                1
## 6     1                1
```

Calculate the accuracy!

```
print(sum(accuracy$train_prediction == churn2$train)/3333 * 100) ## 80% accuracy!
```

```
## [1] 80.43804
```