project-7

Chitra Karki

12/3/2021

1. (Data Preparation) Bring in the data and get familiar with the variables.

a. Take a look at the data. Inspect if there are missing values and, if so, impute them.

```
#install.packages("kernlab")
library(kernlab)
data(spam)
str(spam)
```

```
## 'data.frame':    4601 obs. of  58 variables:
##  $ make             : num  0 0.21 0.06 0 0 0 0 0 0.15 0.06 ...
##  $ address          : num  0.64 0.28 0 0 0 0 0 0 0 0.12 ...
##  $ all              : num  0.64 0.5 0.71 0 0 0 0 0 0.46 0.77 ...
##  $ num3d            : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ our              : num  0.32 0.14 1.23 0.63 0.63 1.85 1.92 1.88 0.61 0.19 ...
##  $ over             : num  0 0.28 0.19 0 0 0 0 0 0 0.32 ...
##  $ remove           : num  0 0.21 0.19 0.31 0.31 0 0 0 0.3 0.38 ...
##  $ internet         : num  0 0.07 0.12 0.63 0.63 1.85 0 1.88 0 0 ...
##  $ order            : num  0 0 0.64 0.31 0.31 0 0 0 0.92 0.06 ...
##  $ mail             : num  0 0.94 0.25 0.63 0.63 0 0.64 0 0.76 0 ...
##  $ receive          : num  0 0.21 0.38 0.31 0.31 0 0.96 0 0.76 0 ...
##  $ will             : num  0.64 0.79 0.45 0.31 0.31 0 1.28 0 0.92 0.64 ...
##  $ people           : num  0 0.65 0.12 0.31 0.31 0 0 0 0 0.25 ...
##  $ report           : num  0 0.21 0 0 0 0 0 0 0 0 ...
##  $ addresses        : num  0 0.14 1.75 0 0 0 0 0 0 0.12 ...
##  $ free             : num  0.32 0.14 0.06 0.31 0.31 0 0.96 0 0 0 ...
##  $ business         : num  0 0.07 0.06 0 0 0 0 0 0 0 ...
##  $ email            : num  1.29 0.28 1.03 0 0 0 0.32 0 0.15 0.12 ...
##  $ you              : num  1.93 3.47 1.36 3.18 3.18 0 3.85 0 1.23 1.67 ...
##  $ credit           : num  0 0 0.32 0 0 0 0 0 3.53 0.06 ...
##  $ your             : num  0.96 1.59 0.51 0.31 0.31 0 0.64 0 2 0.71 ...
##  $ font             : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ num000           : num  0 0.43 1.16 0 0 0 0 0 0 0.19 ...
##  $ money            : num  0 0.43 0.06 0 0 0 0 0 0.15 0 ...
##  $ hp               : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ hpl              : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ george           : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ num650           : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ lab              : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ labs             : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ telnet           : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ num857           : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ data             : num  0 0 0 0 0 0 0 0 0.15 0 ...
##  $ num415           : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ num85            : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ technology       : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ num1999          : num  0 0.07 0 0 0 0 0 0 0 0 ...
##  $ parts            : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ pm               : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ direct           : num  0 0 0.06 0 0 0 0 0 0 0 ...
##  $ cs               : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ meeting          : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ original         : num  0 0 0.12 0 0 0 0 0 0.3 0 ...
##  $ project          : num  0 0 0 0 0 0 0 0 0 0.06 ...
##  $ re               : num  0 0 0.06 0 0 0 0 0 0 0 ...
##  $ edu              : num  0 0 0.06 0 0 0 0 0 0 0 ...
##  $ table            : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ conference       : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ charSemicolon    : num  0 0 0.01 0 0 0 0 0 0 0.04 ...
##  $ charRoundbracket : num  0 0.132 0.143 0.137 0.135 0.223 0.054 0.206 0.271 0.03 ...
##  $ charSquarebracket: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ charExclamation  : num  0.778 0.372 0.276 0.137 0.135 0 0.164 0 0.181 0.244 ...
##  $ charDollar       : num  0 0.18 0.184 0 0 0 0.054 0 0.203 0.081 ...
##  $ charHash         : num  0 0.048 0.01 0 0 0 0 0 0.022 0 ...
##  $ capitalAve       : num  3.76 5.11 9.82 3.54 3.54 ...
##  $ capitalLong      : num  61 101 485 40 40 15 4 11 445 43 ...
##  $ capitalTotal     : num  278 1028 2259 191 191 ...
##  $ type             : Factor w/ 2 levels "nonspam","spam": 2 2 2 2 2 2 2 2 2 2 ...
```

```
dim(spam)
```

```
## [1] 4601   58
```

```
#summary(spam)
sum(is.na(spam))
```

```
## [1] 0
```

no missig vlaues. The variable type is of factor type. There are 4601 observations and 58 feature variables.

b. Explore data using numerical and graphical EDA techniques. For example, what is the percentage of spam emails? What are the types (categorical or continuous) of the inputs? Are there any peculiar features for any variable(s) that we should pay attention to? Do not present any R output for this part unless really necessary. Instead, summarize your ndings in concise language.

```
# percentage of spam and nonspam
mail = table(spam$type)
round(100*mail/sum(mail),2)
```

```
##
## nonspam    spam
##    60.6    39.4
```

The capitlLong and capitalTotal fearures have higher variablity or big range in comparision to other fearures.

c. Randomly divide your datasets into the training sample and for the test sample with a ratio of 2:1. We will use the training sample to train a number of models and then use the test sample to compare them.

```
spam$type = ifelse(spam$type == "spam",1,0)
set.seed(123)
train.index =  sample(1:nrow(spam),size = 2/3 * nrow(spam),replace = F)
train.x = spam[train.index,-ncol(spam)]
train.y = spam[train.index,ncol(spam)]
test.x =  spam[-train.index,-ncol(spam)]
test.y  = as.factor(spam[-train.index,ncol(spam)])
```

2. (Supervised Learning) Try out the following predictive modeling tools. For each method, use the training set to identify the best model and apply the model to the test set. Then plot the ROC curve and compute the C statistic or C index (area under the ROC curve), all based on the test set performance. It would be best, but not required, to have the ROC curves plotted on one gure and compared. Which method gives the highest C index?   Linear discriminant analysis (LDA);

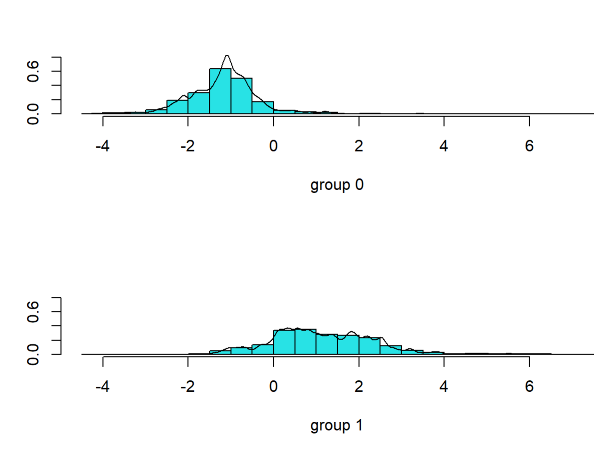
```
library(MASS)
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
model.lda = lda(train.y~.,data = train.x)
plot(model.lda, dimen=1, type="both")
```




```
predict.lda = predict(model.lda,test.x)
table(predict.lda$class,test.y)
```

```
##    test.y
##       0   1
##   0 881 134
##   1  47 472
```

```
roc.lda = roc(test.y~as.numeric(predict.lda$class))
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

There is a certain overlab between two groups.   Train a `best' logistic regression model. Depending on the situation, you might want to use a regularized logistic regression;

```
model.logistic = glm(train.y~.,data = train.x,family = "binomial")
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(model.logistic)
```

```
##
## Call:
## glm(formula = train.y ~ ., family = "binomial", data = train.x)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.2391  -0.2150   0.0000   0.0901   4.8899
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)       -1.770e+00  1.754e-01 -10.091  < 2e-16 ***
## make              -5.020e-01  3.061e-01  -1.640 0.101041
## address           -1.772e-01  1.052e-01  -1.685 0.091988 .
## all                1.244e-01  1.302e-01   0.956 0.339304
## num3d              2.246e+00  2.041e+00   1.100 0.271195
## our                5.650e-01  1.233e-01   4.580 4.64e-06 ***
## over               8.878e-01  3.065e-01   2.897 0.003771 **
## remove             2.384e+00  4.467e-01   5.337 9.45e-08 ***
## internet           5.710e-01  2.435e-01   2.344 0.019059 *
## order              5.228e-01  3.317e-01   1.576 0.115046
## mail               2.610e-01  1.025e-01   2.546 0.010898 *
## receive            1.961e-01  3.610e-01   0.543 0.586953
## will              -8.508e-02  8.598e-02  -0.990 0.322369
## people            -1.343e-01  2.959e-01  -0.454 0.650015
## report             2.004e-01  1.988e-01   1.008 0.313417
## addresses          9.000e-01  7.450e-01   1.208 0.227076
## free               1.418e+00  2.104e-01   6.741 1.58e-11 ***
## business           7.163e-01  2.425e-01   2.954 0.003138 **
## email              8.761e-02  1.478e-01   0.593 0.553244
## you                9.501e-02  4.249e-02   2.236 0.025345 *
## credit             1.418e+00  6.952e-01   2.039 0.041420 *
## your               2.534e-01  6.700e-02   3.782 0.000155 ***
## font               1.523e-01  1.666e-01   0.914 0.360675
## num000             2.335e+00  5.798e-01   4.026 5.66e-05 ***
## money              5.892e-01  2.461e-01   2.394 0.016645 *
## hp                -1.666e+00  2.971e-01  -5.606 2.07e-08 ***
## hpl               -1.126e+00  5.012e-01  -2.246 0.024703 *
## george            -9.548e+00  2.285e+00  -4.178 2.95e-05 ***
## num650             3.439e-01  1.997e-01   1.722 0.085012 .
## lab               -2.266e+00  1.696e+00  -1.336 0.181709
## labs              -1.909e-01  4.386e-01  -0.435 0.663292
## telnet            -1.259e-01  4.109e-01  -0.306 0.759291
## num857             3.243e+00  2.653e+00   1.222 0.221540
## data              -6.520e-01  3.600e-01  -1.811 0.070071 .
## num415             6.694e-01  1.586e+00   0.422 0.672938
## num85             -1.910e+00  8.519e-01  -2.242 0.024955 *
## technology         1.177e+00  3.748e-01   3.140 0.001687 **
## num1999            8.622e-02  2.165e-01   0.398 0.690404
## parts             -7.536e-01  4.309e-01  -1.749 0.080297 .
## pm                -1.010e+00  4.604e-01  -2.193 0.028278 *
## direct            -3.727e-01  4.041e-01  -0.922 0.356379
## cs                -4.948e+01  2.694e+01  -1.837 0.066210 .
## meeting           -2.656e+00  9.528e-01  -2.787 0.005312 **
## original          -9.420e-01  8.030e-01  -1.173 0.240712
## project           -1.503e+00  6.052e-01  -2.483 0.013036 *
## re                -8.078e-01  1.949e-01  -4.144 3.42e-05 ***
## edu               -1.305e+00  3.026e-01  -4.314 1.61e-05 ***
## table             -2.861e+00  2.379e+00  -1.203 0.229142
## conference        -3.216e+00  1.729e+00  -1.860 0.062841 .
## charSemicolon     -1.146e+00  4.975e-01  -2.304 0.021206 *
## charRoundbracket  -2.922e-01  2.931e-01  -0.997 0.318837
## charSquarebracket -1.025e+00  1.311e+00  -0.782 0.434219
## charExclamation    4.756e-01  1.247e-01   3.815 0.000136 ***
## charDollar         5.807e+00  9.203e-01   6.310 2.79e-10 ***
## charHash           2.726e+00  7.329e-01   3.719 0.000200 ***
## capitalAve         2.094e-02  2.125e-02   0.985 0.324424
## capitalLong        4.853e-03  3.004e-03   1.615 0.106274
## capitalTotal       1.126e-03  2.589e-04   4.350 1.36e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4111.7  on 3066  degrees of freedom
## Residual deviance: 1198.4  on 3009  degrees of freedom
## AIC: 1314.4
##
## Number of Fisher Scoring iterations: 13
```

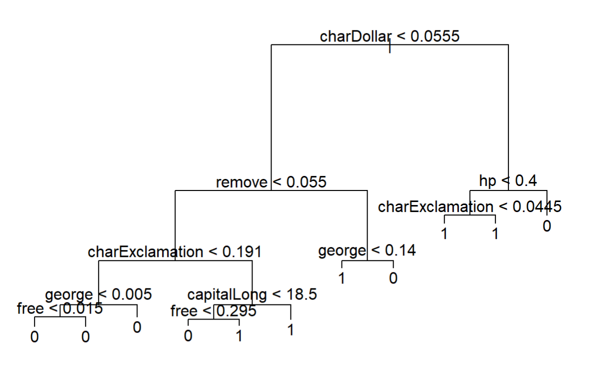
```
predict.logistic = predict(model.logistic,test.x)
roc.logistic = roc(test.y~predict.logistic)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

One single decision tree;

```
library("tree")
set.seed(123)
model.tree = tree(factor(train.y)~.,data = train.x)
plot(model.tree)
text(model.tree,pretty = 0)
```




```
predict.tree = predict(model.tree,test.x,type="class")
roc.tree = roc(test.y~as.numeric(predict.tree))
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

Bagging;

```
set.seed(123)
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
model.bagging = randomForest(train.y~.,data = train.x,mtry=ncol(train.x),importance=T)
```

```
## Warning in randomForest.default(m, y, ...): The response has five or fewer
## unique values. Are you sure you want to do regression?
```

```
predict.bagging = predict(model.bagging,test.x,)
roc.bagging = roc(test.y~predict.bagging)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

Random Forests (RF);

```
set.seed(123)
model.rf = randomForest(train.y~.,data = train.x,mtry = sqrt(ncol(train.x)))
```

```
## Warning in randomForest.default(m, y, ...): The response has five or fewer
## unique values. Are you sure you want to do regression?
```

```
predict.rf = predict(model.rf,test.x)
roc.rf = roc(test.y~predict.rf)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

Boosting.

```
library(gbm)
```

```
## Loaded gbm 2.1.8
```

```
set.seed(123)
model.boosting = gbm(train.y~.,data = train.x)
```

```
## Distribution not specified, assuming bernoulli ...
```

```
predict.boosting = predict(model.boosting,test.x)
```

```
## Using 100 trees...
```

```
roc.boosting = roc(test.y~predict.boosting)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

C.statistics

```
auc(roc.lda);auc(roc.logistic);auc(roc.bagging);auc(roc.rf);auc(roc.boosting);auc(roc.tree)
```

```
## Area under the curve: 0.8641
```

```
## Area under the curve: 0.9747
```

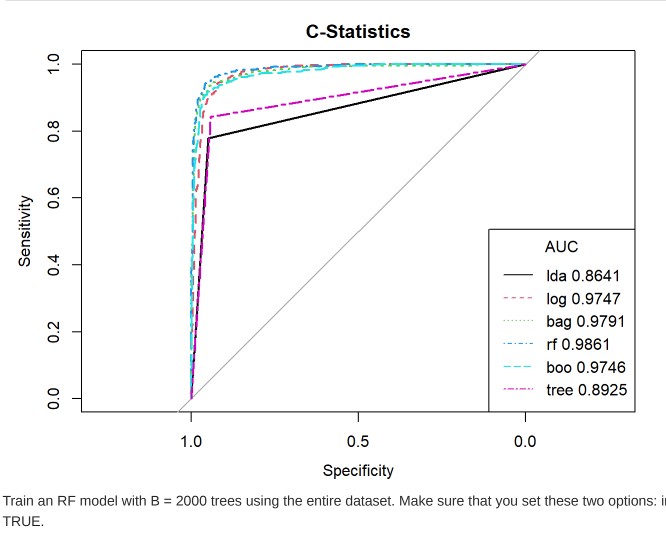
```
## Area under the curve: 0.9791
```

```
## Area under the curve: 0.9861
```

```
## Area under the curve: 0.9746
```

```
## Area under the curve: 0.8925
```

```
plot(roc.lda,col=1,main="C-Statistics",lty=1)
plot(roc.logistic,col=2,lty=2,add=T)
plot(roc.bagging,col=3,lty=3,add=T)
plot(roc.rf,col=4,lty=4,add=T)
plot(roc.boosting,col=5,lty=5,add=T)
plot(roc.tree,col=6,lty=6,add=T)
legend("bottomright",legend = c("lda 0.8641","log 0.9747","bag 0.9791","rf 0.9861", "boo 0.9746", "tree 0.8925"),
lty=c(1:6),col = c(1:6),title="AUC")
```



3. (Additional Features from RF)

Train an RF model with B = 2000 trees using the entire dataset. Make sure that you set these two options: importance = TRUE and proximity = TRUE.

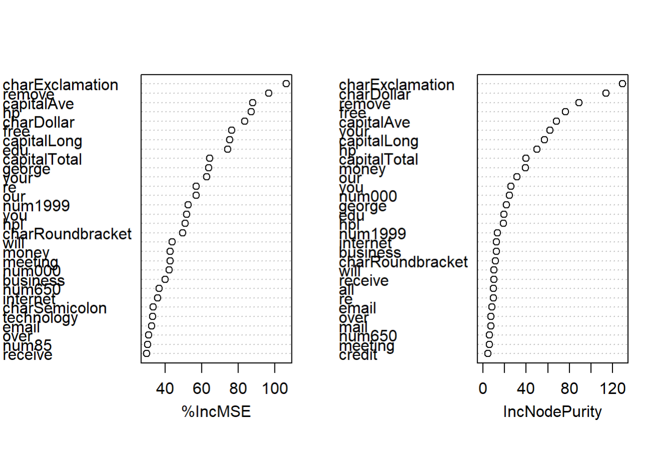
```
set.seed(123)
model.rf.full = randomForest(spam$type~.,data = spam,mtry = sqrt(ncol(train.x)),ntree=2000,importance=T,proximity
=T)
```

```
## Warning in randomForest.default(m, y, ...): The response has five or fewer
## unique values. Are you sure you want to do regression?
```

```
#plot(model.rf.full)
```

a. Obtain the variable importance ranking plots and compare with the variables selected in logistic regression.

```
# varimp logistic vs randomforest
varImpPlot(model.rf.full)
```

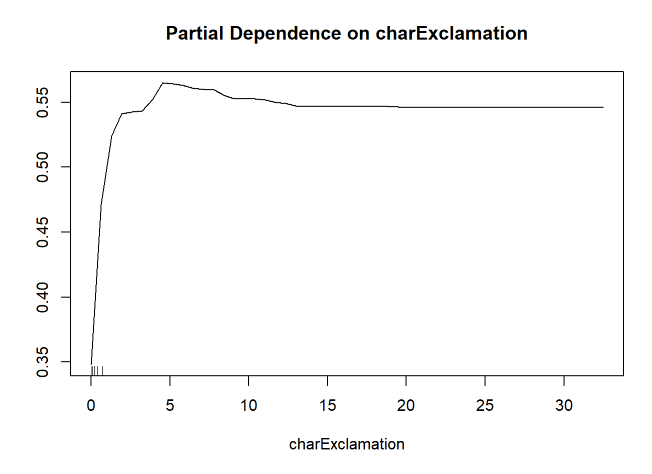



```
imp.logistic = summary(model.logistic)
names(which(imp.logistic$coefficients[,4] > 0.1))
```

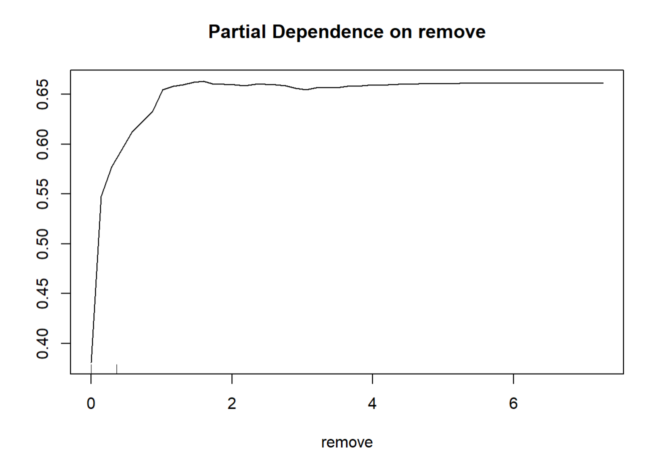
```
##  [1] "make"              "all"               "num3d"
##  [4] "order"             "receive"           "will"
##  [7] "people"            "report"            "addresses"
## [10] "email"             "font"              "lab"
## [13] "labs"              "telnet"            "num857"
## [16] "num415"            "num1999"           "direct"
## [19] "original"          "table"             "charRoundbracket"
## [22] "charSquarebracket" "capitalAve"        "capitalLong"
```

b. Obtain the partial dependence plot for the top two variables that you deem most impor- tant.

```
partialPlot(model.rf.full,spam,charExclamation)
```




```
partialPlot(model.rf.full,spam,remove)
```



(c) Obtain the proximity matrix and transform it into a distance or dissimilarity matrix D. Apply an MDS technique to D and plot the results, by specifying the spam (red) or regular (green) email status with di erent colors. Interpret the

```
proxitimy.matrix = model.rf.full$proximity
library(cluster)
D = daisy(scale(spam),metric = "euclidean")
```

```
## Warning in daisy(scale(spam), metric = "euclidean"): binary variable(s) 58
## treated as interval scaled
```

```
#classical MDS
model.cmd = cmdscale(D,eig = T,k=2)
x = model.cmd$points[,1]
y = model.cmd$points[,2]
plot(x,y,col=ifelse(spam$type==1,"red","green"))
```