Data Science Toolbox 3: Documentation

```
knitr::opts_knit$set(root.dir='../')

library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
library(data.table)
library(ggplot2)
library(ranger)
library(knitr)

set.seed(0xC0FFEE)
```

Load in the data set:

```
columns <- read.table(
    "./data/kddcup.names",
    sep=":",
    skip=1,  # the first column name are the labels, but those are at the end!
    as.is=T
)
column_names <- c(columns[,1], 'label')

connection_events <- read.csv(
    "./data/kddcup.data_10_percent.gz",
    col.names=column_names
)

setDT(connection_events)  # convert from data.frame to data.table
```

K-Fold Validation

```
k_folds <- createFolds(connection_events$label, k=10)
```

This generates 10 sets of indices on the data. These are arranged such that similar amounts of each label are in each set.

To use them for k-fold cross-validation: 1. Pick each fold one at a time. 2. Treat this as the indices of the testing set. 3. Select all other connection events as the training set. 4. Train your model and get your predictions from the test set.

Example below with random forests.

Random Forests

Basically just bagged decision trees? This `ranger` library is supposed to be a "speedy" implementation:

```
training_indices = -k_folds[[1]]
testing_indices = k_folds[[1]]

training_data <- connection_events[training_indices]
testing_data <- connection_events[testing_indices]
```

```
model <- ranger(
    label~duration+service+src_bytes+dst_bytes+protocol_type,
    data=training_data,
    ## automatically selected since the 'label' column is a factor, but
    ## leave this here for clarity:
    classification=TRUE
)
```

```
## Growing trees.. Progress: 97%. Estimated remaining time: 1 seconds.
```

Only took a few minutes! Now, how to use it? Standard R I think (from the examples in the docs), i.e:

```
predictions <- predict(model, data=testing_data)
```

Inspect this object a bit to find out what it looks like:

```
names(predictions)
```

```
## [1] "predictions"              "num.trees"
## [3] "num.independent.variables" "num.samples"
## [5] "treetype"
```

```
summary(predictions$predictions)
```

```
##            back. buffer_overflow.       ftp_write.    guess_passwd.
##              220                3                0                6
##            imap.         ipsweep.            land.      loadmodule.
##                0              126                0                0
##        multihop.         neptune.            nmap.          normal.
##                0            10948                0             9752
##            perl.             phf.             pod.       portsweep.
##                0                0                2               21
##         rootkit.           satan.           smurf.             spy.
##                0               21            28105                0
##        teardrop.     warezclient.     warezmaster.
##               98              100                2
```

Make a confusion matrix to test it out:

```
kable(table(testing_data$label, predictions$predictions))
```

	back.	buffer_overflow.	ftp_write.	guess_passwd.	imap.	ipsweep.	land.	loadmodule.	multihop.	n
back.	220	0	0	0	0	0	0	0	0	
buffer_overflow.	0	3	0	0	0	0	0	0	0	
ftp_write.	0	0	0	0	0	0	0	0	0	
guess_passwd.	0	0	0	6	0	0	0	0	0	
imap.	0	0	0	0	0	0	0	0	0	
ipsweep.	0	0	0	0	0	118	0	0	0	
land.	0	0	0	0	0	0	0	0	0	
loadmodule.	0	0	0	0	0	0	0	0	0	
multihop.	0	0	0	0	0	0	0	0	0	
neptune.	0	0	0	0	0	0	0	0	0	
nmap.	0	0	0	0	0	8	0	0	0	
normal.	0	0	0	0	0	0	0	0	0	
perl.	0	0	0	0	0	0	0	0	0	
phf.	0	0	0	0	0	0	0	0	0	
pod.	0	0	0	0	0	0	0	0	0	
portsweep.	0	0	0	0	0	0	0	0	0	

	back.	buffer_overflow.	ftp_write.	guess_passwd.	imap.	ipsweep.	land.	loadmodule.	multihop.	n
rootkit.	0	0	0	0	0	0	0	0	0	
satan.	0	0	0	0	0	0	0	0	0	
smurf.	0	0	0	0	0	0	0	0	0	
spy.	0	0	0	0	0	0	0	0	0	
teardrop.	0	0	0	0	0	0	0	0	0	
warezclient.	0	0	0	0	0	0	0	0	0	
warezmaster.	0	0	0	0	0	0	0	0	0	

Repeat the above for all 10 folds:

```
confusion_matrices <-lapply(k_folds, function(fold) {  # approx. 10 minutes
    training_indices <- -fold
    testing_indices <- fold

    training_data <- connection_events[training_indices]
    testing_data <- connection_events[testing_indices]

    model <- ranger(
        label~duration+service+src_bytes+dst_bytes+protocol_type,
        data=training_data,
        ## automatically selected since the 'label' column is a factor, but
        ## leave this here for clarity:
        classification=TRUE
    )

    predictions <- predict(model, data=testing_data)

    confusion_matrix <- table(testing_data$label, predictions$predictions)

    return(confusion_matrix)
})
```

```
summary_matrix = Reduce('+', confusion_matrices)
summary_matrix
```

To avoid re-running this everytime, save a copy of the results into a file:

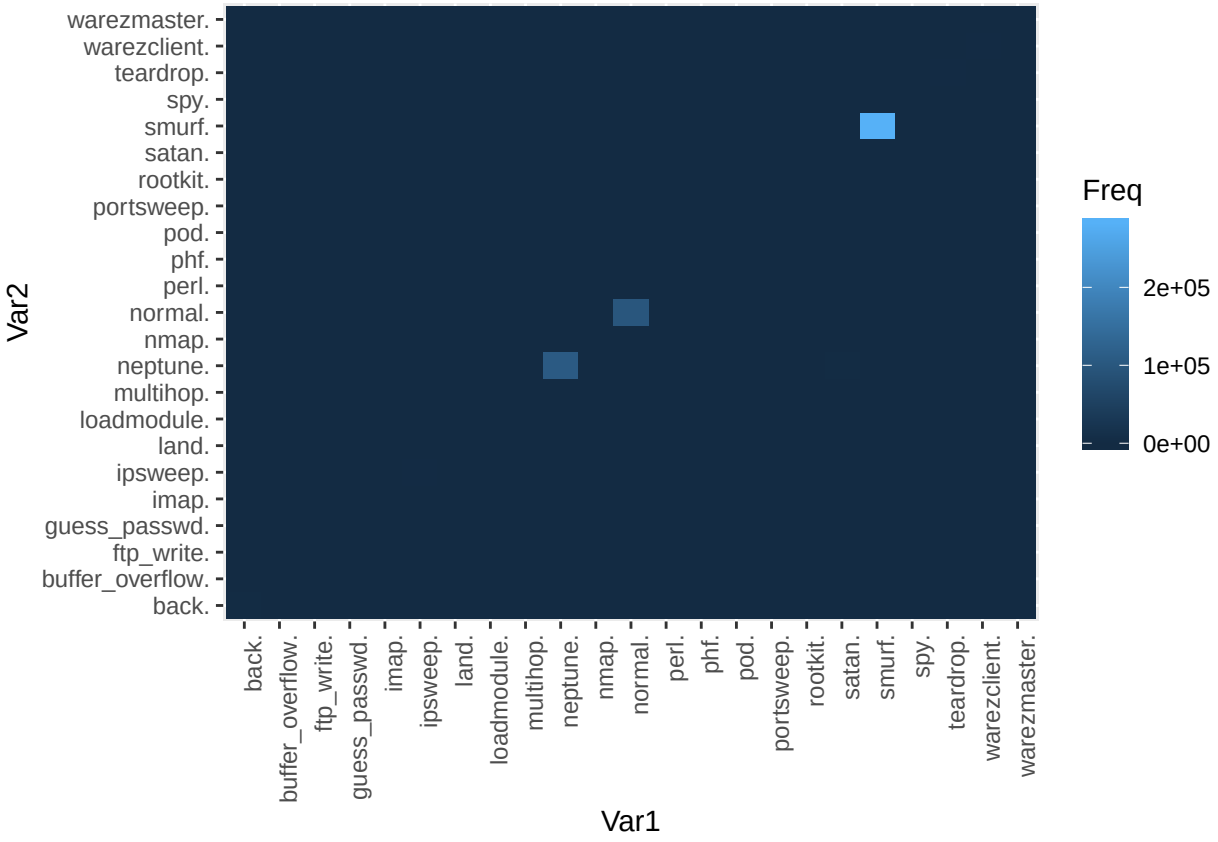
```
saveRDS(
  summary_matrix,
  file='./data/daniel-jones-random-forest-summary-matrix-some-features.rds'
)
```

The load it with:

```
summary_matrix <- readRDS(
  file='./data/daniel-jones-random-forest-summary-matrix-some-features.rds'
)
```

Plot (TODO: needs some tweaking):

```
ggplot(
    data=as.data.frame(summary_matrix),
    aes(x=Var1, y=Var2, fill=Freq),
    limits=c(0, max(summary_matrix))
) + geom_raster() + theme(axis.text.x=element_text(angle=90, hjust=1))
```



Random Forest on all features

Apply the random forest to all features:

```
confusion_matrices <-lapply(k_folds, function(fold) {  # approx. 25 minutes
    training_indices <- -fold
    testing_indices <- fold

    training_data <- connection_events[training_indices]
    testing_data <- connection_events[testing_indices]

    model <- ranger(
        label~.,
        data=training_data,
        ## automatically selected since the 'label' column is a factor, but
        ## leave this here for clarity:
        classification=TRUE
    )

    predictions <- predict(model, data=testing_data)

    confusion_matrix <- table(testing_data$label, predictions$predictions)

    return(confusion_matrix)
})
```

```
summary_matrix = Reduce('+', confusion_matrices)
summary_matrix
```

To avoid re-running this everytime, save a copy of the results into a file:

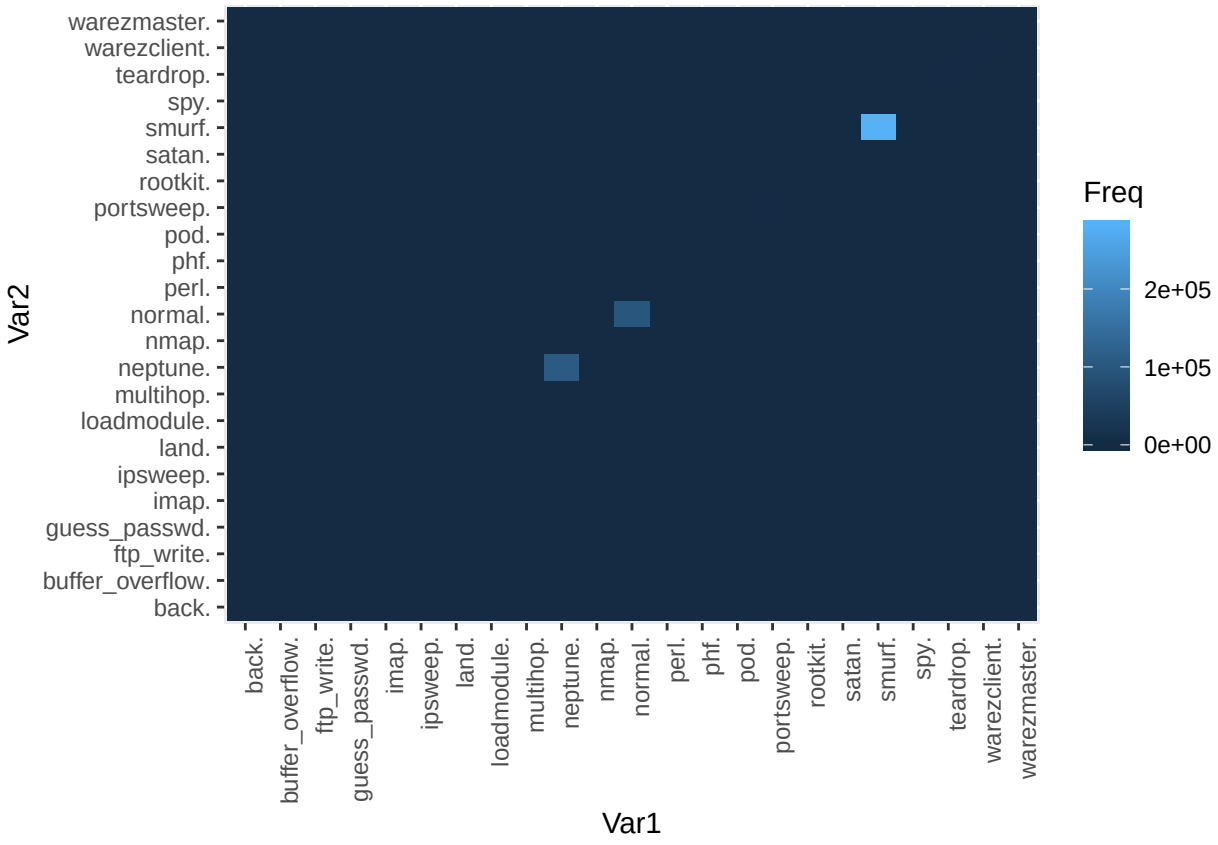
```
saveRDS(
  summary_matrix,
  file='./data/daniel-jones-random-forest-summary-matrix-all-features.rds'
)
```

The load it with:

```
summary_matrix <- readRDS(
  file='./data/daniel-jones-random-forest-summary-matrix-all-features.rds'
)
```

Plot (TODO: needs some tweaking):

```
ggplot(
    data=as.data.frame(summary_matrix),
    aes(x=Var1, y=Var2, fill=Freq),
    limits=c(0, max(summary_matrix))
) + geom_raster() + theme(axis.text.x=element_text(angle=90, hjust=1))
```



TODO

- [x] Should probably run this on all features (I was just lazy and didn't want to wait for it to finish).
- [x] Plot a nice confusion matrix that will help us find out:
- [ ] Where does this model perform the worst?