Machine Learning and Data Analysis

William Seymour

April 14, 2019

1 Opening the terminal and starting R Studio

- Open the terminal by pressing <*Win*> and typing 'terminal'
- Start R by typing 'R' and pressing <*Enter*>

2 Loading the crime data set

First we need to load the packages we need, open the data set in R, and assign it to a variable (in this case *ds*). Then we're going to print out the number of rows in the data set and a summary of the features so we can understand it a little better before we analyse it.

```
library("caret")
library("e1071")
library("randomForest")
library("nnet")

ds = read.csv("/usr/local/practicals/uniq/ml/crime_data.csv")
attach(ds)

nrow(ds)
summary(ds)
```

You'll see a number of different columns here:

1. id: all of the records are sequentially numbered
2. name: the name/gender/age/ethinicity of the person who was arrested
3. sex: the gender of the person who was arrested
4. age: age of the person who was arrested
5. age_cat: age is also seperated into three categories, which might help with some comparisons
6. race: the ethinicity of the person who was arrested
7. juv_fel_count, juv_misd_count: the number of felonies (serious crimes) and misdemeanours (less serious crimes) the person committed while under 18
8. priors_count: the number of prior offences (of any type) the person previously committed while an adult
9. charge_type: whether the crime the person was arrested for was a felony or misdemeanour

10. charge_desc: the crime the person was charged with
11. charge_degree: the degree of the crime from most serious (1st degree felony) to less serious (3rd degree misdemeanour)
12. two_year_recid: whether the person arrested was convicted of another crime within two years

Your goal in this session is to build a model that predicts from a person's record whether they will commit another crime within two years. The following parts of this worksheet will guide you through how to build a few different types of machine learning models to help you do this.

In order to make sure that your models doesn't over fit the data, you'll need to create a training data set that's separate from the data you use to test it. For now, a 75%/25% split will be fine. Don't forget to use the right data sets for training and testing!

```
index = createDataPartition(ds$two_year_recid, p=0.75, list=FALSE)
training = ds[index,]
test = ds[-index,]
```

3 Decision Trees

Because the feature that we're trying to predict is categorical (i.e. we can't order them like we can numbers), we can't use linear regression to help us here. We can, however, use decision trees. Create a decision tree using a selection of features from the data set, evaluate it, and visualise it:

```
model_tree = train(training[,7:9], as.factor(training$two_year_recid), method='rpart')

predict_tree = predict(object=model_tree, test[,7:9])
table(predict_tree)
confusionMatrix(predict_tree, as.factor(test$two_year_recid))

plot(model_tree$finalModel)
text(model_tree$finalModel)
```

You'll notice that we refer to come of the columns by number. You can select a range of columns using a colon (e.g. columns 5, 6, 7, and 8 would be 5:8) and any other selection using a vector (e.g. columns 5, 6, and 8 would be c(5, 6, 8)). To make this easier, the list of features above has the same numbering as the data set in R. Take another look at what the columns represent using the list above and try and pick a selection of columns for your decision tree that you think will best predict recidivism (rather than just copying the example here and using columns 7–9.

4 Random Forests

Random forests are generally more powerful than a single decision tree. Create a random forest using a selection of features from the data set and evaluate it. See if it performs better than the decision tree that you made in the previous exercise.

```
model_forest = train(training[,7:9], as.factor(training$two_year_recid), method='rf')

predict_forest = predict(object=model_tree, test[,5:7])
table(predict_forest)
confusionMatrix(predict_forest, as.factor(test$two_year_recid))

plot(model_forest$finalModel)
```

You'll notice that the output when you plot the forest isn't particularly insightful. This is because the forest is a collection of a large number of trees (for these examples probably several thousand), all with different splits on different combinations of features. You can't visualise them all, and visualising a single tree from the forest wouldn't tell you anything meaningful about the model. C'est la vie.

5 Neural Networks

Mimicing the structure of the brain, neural networks have the potential to be very powerful, but understanding how a particular model works can be very difficult. Create a neural network using a selection of features from the data set, evaluate it, and print out the weights:

```
model_nn = train(training[,7:9], as.factor(training$two_year_recid), method='nnet')

predict_nn = predict(object=model_nn, test[,5:7])
table(predict_nn)
confusionMatrix(predict_nn, as.factor(test$two_year_recid))

summary(model_nn)
```

Once you been waiting a while for R to train your neural network, you might realise that the process can be quite slow (especially on the lab machines). You can speed this up by either reducing the number of features that you include in the model, or by training the model on less data—if you recreate the training and test sets using a different p value then the amount of the data set that gets used to train will be altered accordingly.

6 Challenge: Build the Best Classifier

Now that you've had an opportunity to try out a few different machine learning methods, your challenge for the remainder of this session is to make the best classifier you can that will predict whether someone will re-offend (the two_year_recid value). At the end of the session we'll see who managed to achieve the classifier with the highest accuracy. Obviously, you cannot include the two_year_recid term in your model! This is only for checking whether your predictions were correct.