07_1_Machine_Learning_Labels

Yue Xiong, LMU, yue.xiong@stat.uni-muenchen.de

Introduction

In this tutorial, we'll discuss how to formulate a policy problem or a social science question in the machine learning framework; how to transform raw data into something that can be fed into a model; how to build, evaluate, compare, and select models; and how to reasonably and accurately interpret model results.

This tutorial is adapted according to chapter "Machine Learning" of Big Data and Social Science.

Setup

```
library('dbplyr')  # load the corresponding libraries
library('dplyr')
library('RSQLite')
```

```
# Establish a connection to the ncdoc.db database
database_path = "F:/hiwi_work_notebook/ncdoc.db"
conn = DBI::dbConnect(SQLite(), database_path)
```

```
# Checking the tables saved in this database
src_dbi(conn)
```

```
## src:  sqlite 3.36.0 [F:\hiwi_work_notebook\ncdoc.db]
## tbls: feature_age_2008, feature_age_2013, feature_age_first_admit,
##   feature_agefirstadmit, feature_length_sentence_2000_2008,
##   feature_length_sentence_2000_2013, feature_num_admits_2000_2008,
##   feature_num_admits_2000_2013, features_2000_2008, features_2000_2013, inmate,
##   offender, recidivism_labels_2009_2013, recidivism_labels_2014_2018,
##   sentences, sentences_prep
```

Problem Formulation

Our Machine Learning Problem >Of all prisoners released, we would like to predict who is likely to reenter jail within *5* years of the day we make our prediction. For instance, say it is Jan 1, 2009 and we want to identify which >prisoners are likely to re-enter jail between now and end of 2013. We can run our predictive model and identify who is most likely at risk. The is an example of a *binary classification* problem.

Note the outcome window of 5 years is completely arbitrary. You could use a window of 5, 3, 1 years or 1 day.

In order to predict recidivism, we will be using data from the `inmate` and `sentences` table to create labels (predictors, or independent variables, or X variables) and features (dependent variables, or Y variables).

We need to munge our data into **labels** (1_Machine_Learning_Labels.rmd) and **features** (2_Machine_Learning_Features.rmd) before we can train and evaluate **machine learning models** (3_Machine_Learning_Models.rmd).

Creating Labels (Outcomes)

First, we create a new sentence table `sentences_prep` that includes the sentence begin and end dates in date format.

```
sql_string = "drop table if exists sentences_prep;"
DBI::dbSendStatement(conn, sql_string)
```

```
## <SQLiteResult>
##   SQL  drop table if exists sentences_prep;
##   ROWS Fetched: 0 [complete]
##        Changed: 0
```

```
sql_string = "create table sentences_prep as
select inmate_doc_number,
cast(inmate_sentence_component as integer) as sentence_component,
date([sentence_begin_date_(for_max)]) as sentence_begin_date,
date(actual_sentence_end_date) as sentence_end_date
from sentences;"

DBI::dbSendQuery(conn, sql_string)
```

```
## Warning: Closing open result set, pending rows
```

```
## <SQLiteResult>
##   SQL  create table sentences_prep as
## select inmate_doc_number,
## cast(inmate_sentence_component as integer) as sentence_component,
## date([sentence_begin_date_(for_max)]) as sentence_begin_date,
## date(actual_sentence_end_date) as sentence_end_date
## from sentences;
##   ROWS Fetched: 0 [complete]
##        Changed: 0
```

We then create a table `release_dates_2000_2008`, which is based on the `sentence_prep` table. We take all of the records for `inmate_doc_number` and `sentence_end_date` between 2000 and 2008.

```
sql_query = "drop table if exists release_dates_2000_2008;"
DBI::dbSendStatement(conn, sql_query)
```

```
## Warning: Closing open result set, pending rows
```

```
## <SQLiteResult>
##   SQL  drop table if exists release_dates_2000_2008;
##   ROWS Fetched: 0 [complete]
##        Changed: 0
```

```
sql_query ="create temp table release_dates_2000_2008 as
select inmate_doc_number, sentence_end_date
from sentences_prep
where sentence_end_date >= '2000-01-01' and sentence_end_date <= '2008-12-31';"
DBI::dbSendQuery(conn, sql_query)
```

```
## Warning: Closing open result set, pending rows
```

```
## <SQLiteResult>
##   SQL  create temp table release_dates_2000_2008 as
## select inmate_doc_number, sentence_end_date
## from sentences_prep
## where sentence_end_date >= '2000-01-01' and sentence_end_date <= '2008-12-31';
##   ROWS Fetched: 0 [complete]
##        Changed: 0
```

Next we create a table `last_exit_2000_2008`, which takes the *maximum* (most recent) `sentence_end_date` for every `inmate_doc_number`. This table will only have one entry per `inmate_doc_number`, so for any given `inmate_doc_number`, or individual, we know their *most recent* release year.

```
sql_string = "drop table if exists last_exit_2000_2008;"
DBI::dbSendStatement(conn, sql_string)
```

```
## Warning: Closing open result set, pending rows
```

```
## <SQLiteResult>
##   SQL  drop table if exists last_exit_2000_2008;
##   ROWS Fetched: 0 [complete]
##        Changed: 0
```

```
sql_string ="create temp table last_exit_2000_2008 as
select inmate_doc_number, max(sentence_end_date) sentence_end_date
from release_dates_2000_2008
group by inmate_doc_number;"

DBI::dbSendQuery(conn, sql_string)
```

```
## Warning: Closing open result set, pending rows
```

```
## <SQLiteResult>
##   SQL  create temp table last_exit_2000_2008 as
## select inmate_doc_number, max(sentence_end_date) sentence_end_date
## from release_dates_2000_2008
## group by inmate_doc_number;
##   ROWS Fetched: 0 [complete]
##        Changed: 0
```

We then find everyone admitted into prison between 2009 and 2013 and create table `admit_2009_2013`.

```
sql_string = "drop table if exists admit_2009_2013;"
DBI::dbSendStatement(conn, sql_string)
```

```
## Warning: Closing open result set, pending rows
```

```
## <SQLiteResult>
##   SQL  drop table if exists admit_2009_2013;
##   ROWS Fetched: 0 [complete]
##        Changed: 0
```

```
sql_string ="create temp table admit_2009_2013 as
select inmate_doc_number, sentence_component, sentence_begin_date
from sentences_prep
where sentence_begin_date >= '2009-01-01' and sentence_begin_date <= '2013-12-31' and sentence_componen
"
DBI::dbSendQuery(conn, sql_string)
```

```
## Warning: Closing open result set, pending rows
```

```
## <SQLiteResult>
##   SQL  create temp table admit_2009_2013 as
## select inmate_doc_number, sentence_component, sentence_begin_date
## from sentences_prep
## where sentence_begin_date >= '2009-01-01' and sentence_begin_date <= '2013-12-31' and sentence_compo
##
##   ROWS Fetched: 0 [complete]
##        Changed: 0
```

Next, we do a `left join` on the `last_exit_2000_2008` (left) table and the `admit_2009_2013` (right) table on the `inmate_doc_number` field. The resulting table will keep all the entries from the left table (most recent releases between 2000 and 2008) and add their admits between 2009 and 2013. Now we can create a label: 0 indicates *no recidivism*, 1 indicates *recidivism*, i.e. that person did return to jail between 2009 and 2013.

```
sql_string = "drop table if exists recidivism_2009_2013;"
DBI::dbSendStatement(conn, sql_string)
```

```
## Warning: Closing open result set, pending rows
```

```
## <SQLiteResult>
##   SQL  drop table if exists recidivism_2009_2013;
##   ROWS Fetched: 0 [complete]
##        Changed: 0
```

```
sql_string ="create temp table recidivism_2009_2013 as
select r.inmate_doc_number, r.sentence_end_date, a.sentence_begin_date,
case when a.sentence_begin_date is null then 0 else 1 end recidivism
from last_exit_2000_2008 r
left join admit_2009_2013 a on r.inmate_doc_number = a.inmate_doc_number;"
DBI::dbSendQuery(conn, sql_string)
```

```
## Warning: Closing open result set, pending rows
```

```
## <SQLiteResult>
##   SQL  create temp table recidivism_2009_2013 as
## select r.inmate_doc_number, r.sentence_end_date, a.sentence_begin_date,
## case when a.sentence_begin_date is null then 0 else 1 end recidivism
## from last_exit_2000_2008 r
## left join admit_2009_2013 a on r.inmate_doc_number = a.inmate_doc_number;
##   ROWS Fetched: 0 [complete]
##        Changed: 0
```

We then remove any potential duplicates and create the final label table `recidivism_labels_2009_2013`.

```
sql_string = "drop table if exists recidivism_labels_2009_2013;"
DBI::dbSendStatement(conn, sql_string)
```

```
## Warning: Closing open result set, pending rows
```

```
## <SQLiteResult>
##   SQL  drop table if exists recidivism_labels_2009_2013;
##   ROWS Fetched: 0 [complete]
##        Changed: 0
```

```
sql_string ="create table recidivism_labels_2009_2013 as
select distinct inmate_doc_number, recidivism
from recidivism_2009_2013;"
DBI::dbSendQuery(conn, sql_string)
```

```
## Warning: Closing open result set, pending rows
```

```
## <SQLiteResult>
##   SQL  create table recidivism_labels_2009_2013 as
## select distinct inmate_doc_number, recidivism
## from recidivism_2009_2013;
##   ROWS Fetched: 0 [complete]
##        Changed: 0
```

Finally, we load the label table into `label_2009_2013` and inspect the first observations.

```
sql_string = "SELECT *
              FROM recidivism_labels_2009_2013"
label_2009_2013 = data.frame(tbl(conn, sql(sql_string)))
```

```
## Warning: Closing open result set, pending rows
```

```
head(label_2009_2013, n=5)
```

```
##   INMATE_DOC_NUMBER recidivism
## 1           0000028          0
## 2           0000062          0
## 3           0000114          0
## 4           0000133          0
## 5           0000152          1
```

Following the machine learning pipeline, we will need a second label table for creating a test set later on. To facilitate the label generation process, we define a function called `create_labels` that automates all steps that are needed to create the label table. In essence, it runs all previous steps for a given prediction start date and prediction end date.

```
library("glue")
```

```
## Warning: package 'glue' was built under R version 4.0.5
```

```
##
## Attaching package: 'glue'
```

```
## The following object is masked from 'package:dplyr':
##
##     collapse
```

```
end_year = "2011"
glue("year_{end_year}")
```

```
## year_2011
```

```
create_labels <- function(features_end, prediction_start, prediction_end, conn) {
  # @param features_end
  # @param prediction_start
  # @param prediction_end
  # @param conn: obj

  end_x_year = format(as.Date(features_end, format="%Y-%m-%d"),"%Y")
  start_y_year = format(as.Date(prediction_start, format="%Y-%m-%d"),"%Y")
  end_y_year = format(as.Date(prediction_end, format="%Y-%m-%d"),"%Y")

  drop_script_1 = "drop table if exists sentences_prep;"
  sql_script_1 = glue("create table sentences_prep as
select inmate_doc_number, cast(inmate_sentence_component as integer) as sentence_component, date([sente
date(actual_sentence_end_date) as sentence_end_date
from sentences;")

  drop_script_2 = glue("drop table if exists release_dates_2000_{end_x_year};")
  sql_script_2 = glue("create temp table release_dates_2000_{end_x_year} as
select inmate_doc_number, sentence_end_date
from sentences_prep where sentence_end_date >= '2000-01-01' and sentence_end_date <= '{features_end}';"

  drop_script_3 = glue("drop table if exists last_exit_2000_{end_x_year};")
  sql_script_3 = glue("create temp table last_exit_2000_{end_x_year} as select inmate_doc_number, max(s

  drop_script_4 = glue("drop table if exists admit_{start_y_year}_{end_y_year};")
  sql_script_4 = glue("create temp table admit_{start_y_year}_{end_y_year} as select inmate_doc_number,

  drop_script_5 = glue("drop table if exists recidivism_{start_y_year}_{end_y_year};")
  sql_script_5 = glue("create temp table recidivism_{start_y_year}_{end_y_year} as
select r.inmate_doc_number, r.sentence_end_date, a.sentence_begin_date, case when a.sentence_begin_date
left join admit_{start_y_year}_{end_y_year} a on r.inmate_doc_number = a.inmate_doc_number;")
```

```
  drop_script_6 = glue("drop table if exists recidivism_labels_{start_y_year}_{end_y_year};")
  sql_script_6 = glue("
create table recidivism_labels_{start_y_year}_{end_y_year} as
select distinct inmate_doc_number, recidivism
from recidivism_{start_y_year}_{end_y_year};")

  DBI::dbSendStatement(conn, drop_script_1)
  DBI::dbSendQuery(conn, sql_script_1)

  DBI::dbSendStatement(conn, drop_script_2)
  DBI::dbSendQuery(conn, sql_script_2)

  DBI::dbSendStatement(conn, drop_script_3)
  DBI::dbSendQuery(conn, sql_script_3)

  DBI::dbSendStatement(conn, drop_script_4)
  DBI::dbSendQuery(conn, sql_script_4)

  DBI::dbSendStatement(conn, drop_script_5)
  DBI::dbSendQuery(conn, sql_script_5)

  DBI::dbSendStatement(conn, drop_script_6)
  DBI::dbSendQuery(conn, sql_script_6)

  sql_query = glue("select * from recidivism_labels_{start_y_year}_{end_y_year}")
  df_label = data.frame(tbl(conn, sql(sql_query)))

  return(df_label)
}
```

The `create_labels` function takes a `features_end` date, a `prediction_start` date and a `prediction_end` date as arguments. Our second label table covers recidivism between 2014 and 2018, based on all releases between 2000 and 2013.

```
label_2014_2018 = create_labels('2013-12-31', '2014-01-01', '2018-12-31', conn)
```

```
head(label_2014_2018, n=5)
```

```
##   INMATE_DOC_NUMBER recidivism
## 1           0000028          0
## 2           0000033          0
## 3           0000035          0
## 4           0000037          0
## 5           0000062          0
```

```
DBI::dbDisconnect(conn)
```