Data Preparation

Ammar Hasan 150454388

20 November 2018

Contents

1 Introduction

This report summaries the Data Preparation stage of the CRISP-DM cycle for this project. In particular, this report covers the background of the data and how it was modified for further analysis (selection, cleaning, transforms, etc.).

2 Data Background

The data as stated in the Business Understanding report is collected from a online learning program by Newcastle University on Cyber Security hosted in Future Learn. The collected data on the users is anonymised and is collected for the 7 runs of the program. The data used for this project is derived from the last run for questions responses, archetype survey and step activity tables. The last run was only selected to attempt to limit the differences in field counts and types between tables.

3 Dataset Description

This section describes the dataset that was constructed for some of the Data Understanding phase steps field by field, and with other preprocessing highlights.

3.1 Fields

3.1.1 learner-id

This is the field used to uniquely identify learners from the online program throughout the data (used to help join the cleaned tables together).

3.1.2 archetype

This field is derived from the archetype field from the archetype survey table and is used to represent the type of learner a user is according to their responses.

3.1.3 week-completed-tasks

There are 3 fields of this variation for each week, and they represent the completed tasks for the users in a given week. This is derived from the step activity table.

3.1.4 week-total-marks

There are 3 fields of this variation for each week, and they represent the total marks gained (1 for each correct answer) for all attempts on questions for the users in a given week. This is derived from the question response table.

3.1.5 week-total-attempts

There are 3 fields of this variation for each week, and they represent the total attempts on all questions for the users in a given a week. This is derived from the question response table.

3.2 Table Preprocessing

3.2.1 Archetype Survey Table Cleaning

```
idIndex = 1
responseIndex = 3

archetypesClean = function(df){

  # Remove fields that won't be used (id, responded_at)
  df = df[-responseIndex]
  df = df[-idIndex]

  # fix id format
  df$learner_id = as.character(df$learner_id)

  return(df)
}
```

- The response field and an id field (not the learner id) are removed since they will not be used in future analysis
- The learner id is converted to character format to avoid join issues.

3.2.2 Question Response Table Cleaning

```
clozeIndex = 8
qTypeIndex = 3

questionsClean = function(df){

  # Remove empty cloze response field
  df = df[-clozeIndex]

  # Remove question type field (doesn't change)
  df = df[-qTypeIndex]

  # Remove redudant quiz_number field
  df = df[-stepFieldIndex]

  # Remove rows with id empty fields
  df = drop_na(df, learner_id)

  # fix date format
  df$submitted_at = as.Date(df$submitted_at)

  # fix 'correct' field format
  df$correct = as.logical(df$correct)

  # fix id format
  df$learner_id = as.character(df$learner_id)

  return(df)
}
```

- The cloze response field and the question type field (not the learner id) are removed since one is empty and the other is completely identical for all rows. Redudant quiz number field is also removed.
- The learner id is converted to character format to avoid join issues.
- The date formats are also fixed to date and NAs removed for later use.
- Fields with empty ids removed

3.2.3 Step Activity Table Cleaning

```
stepFieldIndex = 2

stepsClean = function(df){

  # Remove redudant step field
  df = df[-2]

  # Replace empty strings with NA to stop date formatting issues
  df$last_completed_at[df$last_completed_at == ''] = NA

  # fix date format
  df$first_visited_at = as.Date(df$first_visited_at)
  df$last_completed_at = as.Date(df$last_completed_at)

  # fix id format
  df$learner_id = as.character(df$learner_id)

  return(df)
}
```

- The redundant step field is removed since it can be represented by the week number and step number fields.
- The learner id is converted to character format to avoid join issues.
- The date formats are also fixed to date for later use.

3.3 Aggregation

```
# aggregate function used to aggregate both tables
# takes function to carry aggregation by week on a field
agg = function(df, func){

  # add fields for completed steps/marks in all weeks
  df =  left_join(func(df, 1), func(df, 2), by = 'learner_id') %>%
    left_join(., func(df, 3), by = 'learner_id')

  # replace any NAs (dropped off) with 0s
  df[is.na(df)] = 0

  return(df)
}
```

To help with aggregation a generic aggregation function was created to do a left join for the 3 aggregations (on learner id) done by a function passed to it. Also, removes NAs for zeros.

3.3.1 Question Response Table Aggregation

```
# question aggregate just calls the general aggregates and passes
# the function it needs to aggregate marks (# of correct questions) by week
# and number of tries
questionsAgg = function(df){
  agg(df, questionWeekMarks)
}

# function used to aggregate amount of marks (# of correct questions)
# earned each week (not na) and attempts
questionWeekMarks = function(df, week){

  # subset and aggregate (count marks by summing (1 is true) and length for attempts)
  weekDf = subset(df, week_number == week)
  marks = aggregate(weekDf$correct,
                    list(learner_id= weekDf$learner_id),
                    function(x) c(sum = sum(x), n = length(x)))

  # unpack the vector x and rename
  marks[paste('week', week, '_total_marks', sep = '')] =
    marks[,2][,1]
  marks[paste('week', week, '_total_attempts', sep = '')] =
    marks[,2][,2]

  # remove vector and top row
  marks = marks[-2]
  marks = marks[-1,]

  return(marks)
}
```

- A function is constructed to aggregate question by passing a call to generic aggregate with another function which aggregates for a given week.
- The week aggregation functions aggregates using dplyr aggregate function to find total marks (sum) and attempts (length).
- The attempts and marks are unpacked from one another.

3.3.2 Step Activity Table Aggregation

```
# steps aggregate just calls the general aggregates and passes
# the function it needs to aggregate steps compppleted by week
stepsAgg = function(df){
  agg(df, stepsWeekComp)
}

# function used to aggregate amount of steps completed each week (not na)
stepsWeekComp = function(df, week){
  # subset and aggregate (count completed (not NA))
  weekDf = subset(df, week_number == week)
  comps = aggregate(weekDf$last_completed_at,
                    list(learner_id= weekDf$learner_id),
                    function(x) sum(!is.na(x)))

  # rename default x for completed tasks
  colnames(comps) =
    c(colnames(comps)[1],
      paste('week', week, '_completed_steps', sep = ''))

  return(comps)
}
```

- A function is constructed to aggregate steps by passing a call to generic aggregate with another function which aggregates for a given week.
- The week aggregation functions aggregates using dplyr aggregate function to find total steps completed (sum of non NA completed at dates).

3.4 Merge

```
mergeDfs = function(archetypesDf,questionsAggDf, stepsAggDf){

  # join by learner id
  progressByArchetypeDf =
    left_join(stepsAggDf, questionsAggDf, by = 'learner_id')

  # replace nas with 0 for incompleted items
  progressByArchetypeDf[is.na(progressByArchetypeDf)] = 0

  # join by learner id for last df
  progressByArchetypeDf =
    left_join(progressByArchetypeDf, archetypesDf, by = 'learner_id')

  # remove NAs for archetype by replacing with unspecified
  progressByArchetypeDf$archetype =
    fct_explicit_na(progressByArchetypeDf$archetype, 'Unspecified')

  return(progressByArchetypeDf)
}
```

- To finally join all the cleaned and/or aggregated tables this function is called which uses left joins on learner id to join the cleaned and aggregated datasets.
- 0 is used to replace NA attempts/questions/steps and 'Unspecified' is used to replace NA archetypes.

3.5 Munge Preprocessing Script

```
# data-cleaning
archetypesDf = archetypesClean(cyber.security.7.archetype.survey.responses)
questionsDf = questionsClean(cyber.security.7.question.response)
stepsDf = stepsClean(cyber.security.7.step.activity)

# data-aggregation
stepsAggDf = stepsAgg(stepsDf)
questionsAggDf = questionsAgg(questionsDf)

# data-merge
progressTypeDf = mergeDfs(archetypesDf, questionsAggDf, stepsAggDf)
```

This simply calls the appropriate cleaning, aggregation and then merge functions described in this report when ProjectTemplate is loaded.

4 Summary

To summarise, the Data Preparation stage in this project forms a dataset for analysis using cleaning, aggregation and merge functions run on the archetype, questions and step activity datasets. The functions are all ran using a preprocessing script that is loaded when ProjectTemplate is.

5 Second Cycle

5.1 Introduction and New Field

In the second cycle it was decided to introduce country data from the enrolment data table for the 7th run. One new fields was added to the final dataset:

- detected_country: the country that was detected for a users

5.2 New Cleaning Method

```
otherFieldsIndexes = 2:12
enrolClean = function(df){

  # Remove unimportant fields
  df = df[-otherFieldsIndexes]

  # Replace '--' with NA with not detected
  df$detected_country =
    factor(df$detected_country, levels=c(levels(df$detected_country), 'Not Detected'))
  df$detected_country[as.character(df$detected_country) == '--'] = 'Not Detected'

  # Drop unused levels (-- dropped)
  df = droplevels(df)

  # fix id format
  df$learner_id = as.character(df$learner_id)

  return(df)
}
```

- Removes fields that will not be used (gender, employment, etc)
- Replaces - - factor level with Not Detected
- The learner id is converted to character format to avoid join issues.

5.3 New Merge Method

```
countryMergeDfs = function(progressByArchetypeDf, countryDf){

  # join by learner id
  progressByArchetypeDf =
    left_join(progressByArchetypeDf, countryDf, by = 'learner_id')

  # Remove rows with empty fields (single user)
  progressByArchetypeDf = drop_na(progressByArchetypeDf)

  return(progressByArchetypeDf)
}
```

- Similar to other merge (joins cycle 1 final dataset by learner id)
- Drops fields with NA

5.4 Munge Preprocessing Script

```
countryDf = enrolClean(cyber.security.7.enrolments) # clean
countryProgressDf = countryMergeDfs(progressTypeDf, countryDf) # merge
```

Similarily to last time, clean then merge.