MATHS 7107 Data Taming Assignment 5

Ky Phong Mai

2023-04-12

Data Cleaning

1. Read data into R, make sure it is a tibble. Display first 6 rows of dataset

```
pacman::p_load(tidyverse, readr, skimr, tidymodels, themis)
```

```
affairs <- read_csv("affairs.csv") %>% as_tibble()  #make sure it is a tibble
affairs %>%head() #result shows that it is read in correctly
```

```
## # A tibble: 6 x 9
##   affair sex      age    ym child religious education occupation  rate
##    <dbl> <chr>  <dbl> <dbl> <chr>     <dbl>     <dbl>      <dbl> <dbl>
## 1      0 male      37 10    no            3        18          7     4
## 2      0 female    27  4    no            4        14          6     4
## 3      0 female    32 15    yes           1        12          1     4
## 4      0 male      57 15    yes           5        18          6     5
## 5      0 male      22  0.75 no            2        17          6     3
## 6      0 female    32  1.5  no            2        17          5     5
```

The dataset is read in correctly, however, the variable types are not read in correctly

2. What is the outcome variable, and what are the predictors

- Outcome variable: *affair*, an indicator of whether the participant had engaged in an affair
- Predictor variables: the remaining variables including:
  - *sex*, sex of participant
  - *age*, age in years of the participant
  - *ym*, number of years the participant had been married
  - *child*, indicator of whether they have a child
  - *religious*, indicator of how religious are they
  - *education*, years of education
  - *occupation*, job status according to Hollinghead classification
  - *rate*, indicator of how they rate their marriage

3. Skim the data. Is there any missing data? How many observations and variables do we have? Have any variables been read in incorrectly

```
skim_without_charts(affairs) #Use skim_without_charts instead of skim
```

Table 1: Data summary

Name	affairs
Number of rows	601
Number of columns	9
Column type frequency:	
character	2
numeric	7
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
sex	0	1	4	6	0	2	0
child	0	1	2	3	0	2	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
affair	0	1	0.25	0.43	0.00	0	0	0	1
age	0	1	32.49	9.29	17.50	27	32	37	57
ym	0	1	8.18	5.57	0.12	4	7	15	15
religious	0	1	3.12	1.17	1.00	2	3	4	5
education	0	1	16.17	2.40	9.00	14	16	18	20
occupation	0	1	4.19	1.82	1.00	3	5	6	7
rate	0	1	3.93	1.10	1.00	3	4	5	5

- There is no missing data
- There are 601 observations and 9 variables
- The variables have been read in incorrectly, including:
  - **sex** and **child** should be changed from character to *categorical*
  - **affair** should be changed from numeric to *categorical*
  - Note that: **religious**, **education**, **occupation**, **rate** should be *categorical* instead of numeric. However, whether to convert type of these variables depends on how we want to analyze the data.

4. Convert the affair variable to a yes/no response (the function ifelse or case_when will be useful). Change all character variables to factors

```
affairs <- affairs%>%
  mutate(affair = factor(ifelse(affair == "1", "Yes", "No")),
         sex = factor(sex),
         child = factor(child))
affairs %>% head()
```

```
## # A tibble: 6 x 9
##   affair sex      age    ym child religious education occupation  rate
##   <fct>  <fct>  <dbl> <dbl> <fct>     <dbl>     <dbl>      <dbl> <dbl>
## 1 No     male      37 10    no            3        18          7     4
## 2 No     female    27  4    no            4        14          6     4
## 3 No     female    32 15    yes           1        12          1     4
## 4 No     male      57 15    yes           5        18          6     5
## 5 No     male      22  0.75 no            2        17          6     3
## 6 No     female    32  1.5  no            2        17          5     5
```

5. Skim the data again and answer the following

```
skim_without_charts(affairs)
```

Table 4: Data summary

Name	affairs
Number of rows	601
Number of columns	9
Column type frequency:	
factor	3
numeric	6
Group variables	None

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
affair	0	1	FALSE	2	No: 451, Yes: 150
sex	0	1	FALSE	2	fem: 315, mal: 286
child	0	1	FALSE	2	yes: 430, no: 171

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
age	0	1	32.49	9.29	17.50	27	32	37	57
ym	0	1	8.18	5.57	0.12	4	7	15	15
religious	0	1	3.12	1.17	1.00	2	3	4	5
education	0	1	16.17	2.40	9.00	14	16	18	20
occupation	0	1	4.19	1.82	1.00	3	5	6	7
rate	0	1	3.93	1.10	1.00	3	4	5	5

(a) **150** people responded as having had affair. **430** people responded as having children
(b) Mean age of respondents is **32.488**. Mean response on religious scale is **3.116**

Exploratory analysis

1. Proportion of female for those who responded Yes and No to having an affair.

```
affairs %>%
  count(affair, sex) %>%
  group_by(affair) %>%
  mutate (proportion = n/sum(n))
```

```
## # A tibble: 4 x 4
## # Groups:   affair [2]
##   affair sex        n proportion
##   <fct>  <fct>  <int>      <dbl>
## 1 No     female   243      0.539
## 2 No     male     208      0.461
## 3 Yes    female    72      0.48
## 4 Yes    male      78      0.52
```

- Of the participants who responded "No" to an affair, 53.880% are female
- Of the participants who responded "Yes" to an affair, 48% are female

It does not appear to have a difference in the proportion of females who will have an affair and those who will not

2. Proportion of having children for those who responded Yes and No to having an affair

```
affairs %>%
  count(affair, child) %>%
  group_by(affair) %>%
  mutate(proportion = n/sum(n))
```

```
## # A tibble: 4 x 4
## # Groups:   affair [2]
##   affair child     n proportion
##   <fct>  <fct> <int>      <dbl>
## 1 No     no      144      0.319
## 2 No     yes     307      0.681
## 3 Yes    no       27      0.18
## 4 Yes    yes     123      0.82
```

- 82% of participants who responded "yes" to having an affair had children
- 68.071% of participants who responded "no" to having an affair had children

Based on this, if you have an affair, you are more likely to have children. However, at this stage, we can't draw any conclusion on causality.

Split and preprocess

1. Using initial_split, create an rsplit of the affairs data

```
set.seed(1234)
affairs_split <- initial_split (affairs)
affairs_split
```

```
## <Training/Testing/Total>
## <450/151/601>
```

- 450 observations in training set
- 151 observations in testing set

2. Use the functions training and testing to obtain the test and training sets. Display first 6 rows of training set

```
affairs_train <- training(affairs_split)
affairs_test <- testing (affairs_split)
affairs_train %>% head()
```

```
## # A tibble: 6 x 9
##   affair sex      age     ym child religious education occupation  rate
##   <fct>  <fct>  <dbl>  <dbl> <fct>     <dbl>     <dbl>      <dbl> <dbl>
## 1 No     female    42 15     yes           3        14          1     3
## 2 No     female    27 10     yes           5        14          1     5
## 3 No     male      22  1.5   no            2        18          5     3
## 4 No     male      37 10     yes           1        16          6     4
## 5 No     female    22  0.125 no            4        12          4     5
## 6 No     male      32  4     no            1        20          6     5
```

3. Purpose of step_downsample from themis package

- step_downsample is one of the preprocessing step that removes rows of data to make occurrence of levels for a factor variable equal. In other words, for a specific factor variable, this technique will remove observations of the majority level to match with the number of observations of the minority level. Thus, the number of observations for each level in a factor variable will be the same. step_downsample creates a specification of a recipe step.
- We want to down sample our data to tackle the issue of class imbalance. If the number of classes are imbalanced, the model will not be able to learn enough from the minority class as it will spend most of its time learning from majority class. In other words, the predictive power for minority class will be very low as compared to the majority class. The model is biased towards the class with a large number of training observations. There are other techniques to handle class imbalance issue as well and it depends on nature of the data set and the model we to fit the data to decide on which method we should use in each case.

4. Create a recipe, based off of our training data:

```
affairs_recipe <- recipe(affair ~ ., data = affairs_train) %>%
  themis::step_downsample(affair) %>%
  step_dummy (all_factor_predictors()) %>%
  step_normalize(all_predictors()) %>%
  prep()

affairs_recipe
```

```
##

## -- Recipe ----------------------------------------------------------------------

##

## -- Inputs

## Number of variables by role

## outcome:   1
## predictor: 8

##

## -- Training information

## Training data contained 450 data points and no incomplete rows.

##

## -- Operations

## * Down-sampling based on: affair | Trained

## * Dummy variables from: sex, child | Trained

## * Centering and scaling for: age, ym, religious, education, ... | Trained
```

5. Complete the following:

(a) Use the function juice (on the recipe) to get your preprocessed training set

```
affairs_train_preprocess <- juice(affairs_recipe)
affairs_train_preprocess %>% head()
```

```
## # A tibble: 6 x 9
##        age     ym religious education occupation  rate affair sex_male child_yes
##      <dbl>  <dbl>     <dbl>     <dbl>      <dbl> <dbl> <fct>     <dbl>     <dbl>
## 1 -1.15    -1.22    -0.848    -0.0653      0.432 1.07  No       -0.932    -1.68
## 2 -0.570   -0.767   -1.72     -0.0653      0.432 1.07  No       -0.932     0.593
## 3 -0.570   -1.22     0.0262    0.348       0.432 1.07  No       -0.932    -1.68
## 4  1.16     1.25     1.77     -0.891       0.432 0.204 No       -0.932     0.593
## 5 -0.570   -0.218    0.900    -0.891      -1.77  0.204 No       -0.932     0.593
## 6  0.00665 -0.218   -1.72      0.760       0.982 0.204 No        1.07      0.593
```

(b) Use the function bake (on the recipe and testing split) to get your preprocessed testing set. This can be both be done in the one function.

```
affairs_test_preprocess <- bake (affairs_recipe, affairs_test)
affairs_test_preprocess %>% head()
```

```
## # A tibble: 6 x 9
##        age     ym religious education occupation   rate affair sex_male
##      <dbl>  <dbl>     <dbl>     <dbl>      <dbl>  <dbl> <fct>     <dbl>
## 1  0.00665  1.25     -1.72    -1.72       -1.77   0.204 No       -0.932
## 2 -1.15    -1.36     -0.848    0.348       0.982 -0.663 No        1.07
## 3 -1.15    -1.36     -0.848   -1.72       -1.77  -0.663 No       -0.932
## 4  2.89     1.25     -0.848   -0.891      -0.117  0.204 No        1.07
## 5  0.00665  1.25      0.900   -0.0653     -1.77  -1.53  No       -0.932
## 6 -0.570    0.332    -0.848   -0.891      -1.77   1.07  No       -0.932
## # i 1 more variable: child_yes <dbl>
```

6. Skim the preprocessed training data. Explain if the 3 preprocessing steps have done what you expect.

```
skim_without_charts(affairs_train_preprocess)
```

Table 7: Data summary

Name	affairs_train_preprocess
Number of rows	234
Number of columns	9
Column type frequency:	
factor	1
numeric	8
Group variables	None

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
affair	0	1	FALSE	2	No: 117, Yes: 117

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
age	0	1	0	1	-1.67	-0.57	0.01	0.58	2.89
ym	0	1	0	1	-1.48	-0.77	-0.22	1.25	1.25
religious	0	1	0	1	-1.72	-0.85	0.03	0.90	1.77
education	0	1	0	1	-2.96	-0.89	-0.07	0.76	1.59
occupation	0	1	0	1	-1.77	-0.67	0.43	0.98	1.53
rate	0	1	0	1	-2.40	-0.66	0.20	1.07	1.07
sex_male	0	1	0	1	-0.93	-0.93	-0.93	1.07	1.07
child_yes	0	1	0	1	-1.68	-1.68	0.59	0.59	0.59

The 3 preprocessing steps have done what we expect. From the table above, we can see that the number of Yes and No for our outcome variable (**affair**) are balanced at 117 now. There are no more categorical variables except for our outcome variable. The categorical predictors **sex** and **child** were converted to dummy variables and are presented as **sex_male** and **child_yes**. All of the predictors are also normalized with mean of 0 and sd of 1.

Tune and fit a model

1. Make a model specification for a k-nearest neighbors model. In the model specification, define that we would like to tune() the neighbors parameter

```
knn_spec <- nearest_neighbor(mode = "classification", neighbors = tune()) %>%
  set_engine("kknn")
```

2. Create a 5-fold cross validation set from the preprocessed training data. Be sure to set a seed for reproducibility using set.seed(1234)

```
set.seed(1234)
affairs_cv <- vfold_cv(data = affairs_train_preprocess, v = 5)
```

3. Use grid_regular to make a grid of k-values to tune our model on. Using levels get 25 unique values for k. You also need to set your neighbors to range from 5 to 75

```
params_grid <- grid_regular (neighbors(range = c(5,75)), levels = 25)
params_grid
```

```
## # A tibble: 25 x 1
##    neighbors
##        <int>
##  1         5
##  2         7
##  3        10
##  4        13
```

```
##  5        16
##  6        19
##  7        22
##  8        25
##  9        28
## 10        31
## # i 15 more rows
```

4. Use tune_grid to tune your k-nearest neighbours model using your cross validation sets and grid of k-values

```
knn_tuned <- tune_grid (object = knn_spec,
                        preprocessor = recipe(affair ~., data = affairs_train_preprocess),
                        resamples = affairs_cv,
                        grid = params_grid)
```

5. What is the value of k that gives the best accuracy based on our tuned model? (Hint: the function select_best will be useful with tuned model as the first parameter and "accuracy" as the second parameter)

```
best_accuracy <- select_best (knn_tuned, "accuracy")
best_accuracy
```

```
## # A tibble: 1 x 2
##   neighbors .config
##       <int> <chr>
## 1        37 Preprocessor1_Model12
```

k = 37 will give the best accuracy based on the tuned model

6. Finalise the k-nearest model using your results from question 6. Print the model specification to make sure it worked. (Hint: the using finalize_model() function is useful here)

```
affairs_final_knn <- finalize_model (knn_spec, best_accuracy)
affairs_final_knn
```

```
## K-Nearest Neighbor Model Specification (classification)
##
## Main Arguments:
##   neighbors = 37
##
## Computational engine: kknn
```

7. Fit your finalised model to the preprocessed training data and save it with the variable name affairs_knn

```
affairs_knn <- affairs_final_knn %>%
  fit(affair~., data = affairs_train_preprocess)

affairs_knn
```

```
## parsnip model object
##
##
## Call:
## kknn::train.kknn(formula = affair ~ ., data = data, ks = min_rows(37L,     data, 5))
##
## Type of response variable: nominal
## Minimal misclassification: 0.3974359
## Best kernel: optimal
## Best k: 37
```

Evaluation

1. Obtain class predictions using your finalised model from the preprocessed test set using predict. Print the first 6 rows to make sure it worked.

```
affairs_test_pred <- predict(affairs_knn, new_data = affairs_test_preprocess)
affairs_test_pred %>% head()
```

```
## # A tibble: 6 x 1
##   .pred_class
##   <fct>
## 1 No
## 2 No
## 3 No
## 4 Yes
## 5 Yes
## 6 No
```

2. Add the true value of affair from the testing data to your predictions (Hint: you could use bind_cols(select(preprocessed_test_data, affair)). You will need to change the variable names. Print the first 6 rows to make sure this worked

```
affairs_test_pred<- affairs_test_pred %>% rename(affair_pred = .pred_class) %>%
  bind_cols(affairs_test_preprocess %>%
              select(affair))
affairs_test_pred %>% head()
```

```
## # A tibble: 6 x 2
##   affair_pred affair
##   <fct>       <fct>
## 1 No          No
```

```
## 2 No          No
## 3 No          No
## 4 Yes         No
## 5 Yes         No
## 6 No          No
```

3. Get a confusion matrix

```
affairs_test_pred %>%
  conf_mat (truth = affair, estimate = affair_pred)
```

```
##           Truth
## Prediction No Yes
##        No  81  11
##        Yes 37  22
```

4. From your confusion matrix, calculate the sensitivity and specificity of your model. Interpret these values in context

```
categorical_metrics <- metric_set(sensitivity, specificity)
affairs_test_pred %>%
  categorical_metrics(
    truth = affair,
    estimate = affair_pred
  )
```

```
## # A tibble: 2 x 3
##   .metric     .estimator .estimate
##   <chr>       <chr>          <dbl>
## 1 sensitivity binary         0.686
## 2 specificity binary         0.667
```

- sensitivity is 68.644%. This means that, 68.644% of those who do not have an affair have been correctly classified as not having an affair
- specificity is 66.667%. This means that, 66.667% of those who have an affair have been correctly classified as having an affair

5. I have a friend: let's call him Bono. Bono is a large alpha male from Liverpool. He is 47 years old, has been married for 15 years and has no children. He places his religious beliefs at a 2, his occupation at a 6, his education at a 20, and he rates his marriage at an astounding 5.

(a) Make a tibble containing Bono's information.

```
bono <- tibble (sex = factor('male'),
                age = 47,
                ym = 15,
                child = factor('no'),
                religious = 2,
```

```
                occupation = 6,
                education = 20,
                rate = 5)

bono
```

```
## # A tibble: 1 x 8
##   sex     age    ym child religious occupation education  rate
##   <fct> <dbl> <dbl> <fct>     <dbl>      <dbl>     <dbl> <dbl>
## 1 male     47    15 no            2          6        20     5
```

(b) Use bake to preprocess Bono's information with the recipe

```
bono_preprocess <- bake (affairs_recipe, bono)
bono_preprocess
```

```
## # A tibble: 1 x 8
##     age    ym religious education occupation  rate sex_male child_yes
##   <dbl> <dbl>     <dbl>     <dbl>      <dbl> <dbl>    <dbl>     <dbl>
## 1  1.74  1.25    -0.848      1.59      0.982  1.07     1.07     -1.68
```

(c) Using the predict() function, obtain a predicted probability (i.e. with type = "prob") that Bono will have an affair

```
predict(affairs_knn, new_data = bono_preprocess, type = "prob")
```

```
## # A tibble: 1 x 2
##   .pred_No .pred_Yes
##      <dbl>     <dbl>
## 1    0.375     0.625
```

Based on the model, it is predicted that there is 62.55% chance that Bono will have an affair

(d) Given the model, I would not be comfortable going to Bono's partner with my prediction. As the sensitivity of the model is only 68.6% on the test data, this means that if we use the model for prediction, 31.4% of people who do not have an affair are incorrectly classified as having an affair. Because 31.4% is a very high percentage given this being a very sensitive and personal topic, it is not wise to jump to the conclusion based on the prediction.