MATHS 7107 Data Taming Assigmment 05

Possakorn Kittipipatthanapong (a1873765)

2023-04-12

Objective for this assignment

The purpose of this assessment is to go through all the necessary stages involved in creating a forecast model using a particular dataset. The process involves tasks such as removing irrelevant data, analyzing the data, preparing the data for modeling, building and refining the model, and eventually assessing the model's accuracy and making predictions based on new data. This evaluation will test a range of abilities that have been acquired during the course.

The data we are looking at today concerns extramarital activities of readers of Psychology Today in 1969. Our aim is to build a predictive model that will determine whether an individual is likely to engage in extramarital affairs based on various aspects of their lives. The variables in this dataset are:

- affair - An indicator of whether the participant had engaged in an affair, categorical.
- sex - the sex of the participant, categorical.
- age - the age in years of the participant, continuous.
- ym - the number of years the participant had been married, continuous.
- child - do they have a child? Categorical.
- religious - how religious are they, ranging from 1 = anti-religious to 5 = very religious.
- education - years of education, ranging from 9 = primary school to 20 = PhD.
- occupation - Job status, ranging from 1-7 according to the Hollinghead classification (reverse numbering so 7 is a better level job).
- rate - How do they rate their marriage, ranging from 1 = unhappy to 5 = very happy.

Data Cleaning

01 Read the data into R, making sure it is a tibble. Display the first 6 rows of the dataset to make sure it has read in correctly

```
df <- read.csv("sc/affairs.csv") %>%
  as.tibble()

kable(head(df), caption = "Display the first 6 rows of the affair dataset") %>%
  kable_styling(latex_options = "hold_position")
```

Table 1: Display the first 6 rows of the affair dataset

affair	sex	age	ym	child	religious	education	occupation	rate
0	male	37	10.00	no	3	18	7	4
0	female	27	4.00	no	4	14	6	4
0	female	32	15.00	yes	1	12	1	4
0	male	57	15.00	yes	5	18	6	5
0	male	22	0.75	no	2	17	6	3
0	female	32	1.50	no	2	17	5	5

02 What is the outcome variable, and what are the predictor variables?

The variables in this model following;
- outcome variable: affair
- predictor variables: sex, age, ym, child, religious, education, occupation, rate

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

```
skim_df <- df %>%
  skim()
print(skim_df)
```

```
## -- Data Summary ------------------------
##                            Values
## Name                       Piped data
## 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
## 1 sex                   0             1   4   6     0        2          0
## 2 child                 0             1   2   3     0        2          0
##
## -- Variable type: numeric ------------------------------------------------------
##   skim_variable n_missing complete_rate   mean    sd     p0 p25 p50 p75 p100
## 1 affair                0             1  0.250 0.433  0       0   0   0    1
## 2 age                   0             1 32.5   9.29  17.5    27  32  37   57
## 3 ym                    0             1  8.18  5.57   0.125   4   7  15   15
## 4 religious             0             1  3.12  1.17   1       2   3   4    5
## 5 education             0             1 16.2   2.40   9      14  16  18   20
## 6 occupation            0             1  4.19  1.82   1       3   5   6    7
## 7 rate                  0             1  3.93  1.10   1       3   4   5    5
##   hist
## 1
## 2
## 3
## 4
## 5
## 6
## 7
```

Following skim function, we could interpret following;
- There are no missing data.
- There are 601 observations and 9 variables. - There are column: affair, column: sex, and column: child been read in incorrectly as numeric, character, and character respectively.

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

```
df <- df %>%
  mutate(affair = as.factor(ifelse(affair=="1","yes","no")),
         sex = as.factor(sex),
         child = as.factor(child)
         )
kable(head(df), caption = "Display the first 6 rows of the affair dataset after type conversio
  kable_styling(latex_options = "hold_position")
```

Table 2: Display the first 6 rows of the affair dataset after type conversion

affair	sex	age	ym	child	religious	education	occupation	rate
no	male	37	10.00	no	3	18	7	4
no	female	27	4.00	no	4	14	6	4
no	female	32	15.00	yes	1	12	1	4
no	male	57	15.00	yes	5	18	6	5
no	male	22	0.75	no	2	17	6	3
no	female	32	1.50	no	2	17	5	5

05 Skim the data again and answer the following.

```
skim_df <- df %>%
  skim()
print(skim_df)
```

```
## -- Data Summary ------------------------
##                            Values
## Name                       Piped data
## 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
## 1 affair                0             1 FALSE          2 no: 451, yes: 150
## 2 sex                   0             1 FALSE          2 fem: 315, mal: 286
## 3 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 hist
## 1 age                   0             1 32.5   9.29 17.5    27  32  37   57
## 2 ym                    0             1  8.18 5.57  0.125   4   7  15   15
## 3 religious             0             1  3.12 1.17  1       2   3   4    5
## 4 education             0             1 16.2   2.40  9      14  16  18   20
## 5 occupation            0             1  4.19 1.82  1       3   5   6    7
## 6 rate                  0             1  3.93 1.10  1       3   4   5    5
```

Following the questions;
- There are 150 people responded as having had an affair. And, there are 430 people responded to having children.
- The mean age of respondents equal to 32.488 years and the mean response on the religious scale is 3.116.

Exploratory analysis

This section is concerned with the exploratory analysis of the data. Exploratory analysis is an important part of any model building process. This is where we will examine possible relationships in our data that may help inform our model. We will look at some of the relationships in our dataset using summary statistics and data visualization.

01 Of the participants who responded "no" to an affair, what proportion of them are female? How about for those who responded "yes" to having an affair? Does there appear to be a difference in the proportion of females who will have an affair opposed to those who will not? (Hint: the function count will be useful for this).

```
df_affair_sex <- df %>% group_by(affair) %>%
  summarize( percent_female = round(sum(sex=="female") / n(), 3) )

kable(df_affair_sex, caption = "Display participants related to having an affair and proportio
  kable_styling(latex_options = "hold_position")
```

Table 3: Display participants related to having an affair and proportion of them are female

affair	percent_female
no	0.539
yes	0.480

The proportion of who responded "no" to affair and be female is 0.539. However, proportion of who responded "yes" to affair and be female is lower as 0.480. There appear to be a difference in the proportion of females who will have an affair opposed to those who will not.

02 What proportion of participants who responded "yes" to having an affair had children? How about those participants who responded "no"? Based on this, are you more likely to have children if you have an affair?

```
df_affair_child <- df %>% group_by(affair) %>%
  summarize( percent_have_children = round(sum(child=="yes") / n(), 3) )

kable(df_affair_child, caption = "Display participants related to having an affair and proporti
  kable_styling(latex_options = "hold_position")
```

Table 4: Display participants related to having an affair and proportion of them had children

affair	percent_have_children
no	0.681
yes	0.820

The proportion of who responded "yes" to having an affair had children is 0.820. However, responded "no" to having an affair had children is lower as 0.681. Based on this, there are statistic evidences to be more likely to have children if you have an affair.

Split and preprocess

In this section, we will look at data splitting and preprocessing in **TidyModels**. Data slitting is an important step in the model building process that will help with avoiding over fitting and the evaluation of models. Preprocessing is the generalised term for performing mathematical operations on your data before modelling it to improve the predictive power of the model.

01 Using initial_split, create an rsplit of the affairs data. How many observations are in the training set and how many are in the testing set? Do not forget to set a seed for reproducibility using set.seed(1234).

```
set.seed(1234)
affair_split <- initial_split(df)
affair_split
```

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

After splitting by initial_split, amount of training set and testing set equal to 450 and 151 datasets, respectively.

02 Use the functions training and testing to obtain the test and training sets. Display the first 6 rows of the training set to make sure this has worked properly.

```
affair_train <- training(affair_split)
affair_test <- testing(affair_split)

kable(head(affair_train), caption = "Display the first 6 rows of the training affair dataset to
  kable_styling(latex_options = "hold_position")
```

Table 5: Display the first 6 rows of the training affair dataset to make sure this has worked properly

affair	sex	age	ym	child	religious	education	occupation	rate
no	female	42	15.000	yes	3	14	1	3
no	female	27	10.000	yes	5	14	1	5
no	male	22	1.500	no	2	18	5	3
no	male	37	10.000	yes	1	16	6	4
no	female	22	0.125	no	4	12	4	5
no	male	32	4.000	no	1	20	6	5

03 What does step_downsample from the themis package do? Why might we want to down sample our data?

themis::step_downsample() is the function that creates from the recipe step to remove rows from the data set. It uses for adjusting balance of each specific factor or response variables to reduce the bias from imbalance factor.

04 In tutorial 3 we saw how to use recipes. Create a recipe, based off of our training data, that will:

```
affair_recipe <- recipe(affair ~ ., data = affair_train) %>%
  step_downsample(affair) %>% # Down sample our data on affair
  step_dummy( all_factor_predictors() ) %>% # Convert all our categorical predictors to dummy
  step_normalize( all_predictors() ) %>% # Normalise all of our predictor
  prep() # Print out the recipe
affair_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
```

05 Complete the following:

```
# Use the function juice (on the recipe) to get your preprocessed training set.
affair_train_preprocess <- juice( affair_recipe )

kable(head(affair_train_preprocess), caption = "Display the first 6 rows of the training affai
  kable_styling(latex_options = "hold_position")
```

```
# Use the function bake (on the recipe and testing split)
affair_test_preprocess <- bake( affair_recipe, affair_test)
kable(head(affair_test_preprocess), caption = "Display the first 6 rows of the testing affair
  kable_styling(latex_options = "hold_position")
```

Table 6: Display the first 6 rows of the training affair dataset after preprocess by juices

age	ym	religious	education	occupation	rate	affair	sex_male	child_yes
-1.1462374	-1.2248889	-0.8481783	-0.0652763	0.432179	1.0706024	no	-0.9318119	-1.6804599
-0.5697930	-0.7669868	-1.7225120	-0.0652763	0.432179	1.0706024	no	-0.9318119	0.5925321
-0.5697930	-1.2248889	0.0261553	0.3475524	0.432179	1.0706024	no	-0.9318119	-1.6804599
1.1595399	1.2477825	1.7748225	-0.8909339	0.432179	0.2037479	no	-0.9318119	0.5925321
-0.5697930	-0.2175043	0.9004889	-0.8909339	-1.766297	0.2037479	no	-0.9318119	0.5925321
0.0066513	-0.2175043	-1.7225120	0.7603812	0.981798	0.2037479	no	1.0685917	0.5925321

Table 7: Display the first 6 rows of the testing affair dataset applied preprocess by bake

age	ym	religious	education	occupation	rate	affair	sex_male	child_yes
0.0066513	1.2477825	-1.7225120	-1.7165914	-1.766297	0.2037479	no	-0.9318119	0.5925321
-1.1462374	-1.3622595	-0.8481783	0.3475524	0.981798	-0.6631067	no	1.0685917	-1.6804599
-1.1462374	-1.3622595	-0.8481783	-1.7165914	-1.766297	-0.6631067	no	-0.9318119	-1.6804599
2.8888729	1.2477825	-0.8481783	-0.8909339	-0.117440	0.2037479	no	1.0685917	0.5925321
0.0066513	1.2477825	0.9004889	-0.0652763	-1.766297	-1.5299612	no	-0.9318119	0.5925321
-0.5697930	0.3319782	-0.8481783	-0.8909339	-1.766297	1.0706024	no	-0.9318119	0.5925321

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

```
affair_train_preprocess_skim <- affair_train_preprocess %>%
  skim()
print(affair_train_preprocess_skim)
```

```
## -- Data Summary ------------------------
##                            Values
## Name                       Piped data
## 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
## 1 affair                0             1 FALSE          2 no: 117, yes: 117
##
## -- Variable type: numeric ------------------------------------------------------
##   skim_variable n_missing complete_rate      mean sd     p0    p25      p50
## 1 age                   0             1 -1.23e-16  1 -1.67  -0.570  0.00665
## 2 ym                     0             1 -1.05e-16  1 -1.48  -0.767 -0.218
## 3 religious              0             1 -9.16e-17  1 -1.72  -0.848  0.0262
## 4 education              0             1 -4.84e-16  1 -2.96  -0.891 -0.0653
## 5 occupation             0             1 -9.70e-17  1 -1.77  -0.667  0.432
## 6 rate                   0             1  2.78e-17  1 -2.40  -0.663  0.204
## 7 sex_male               0             1 -4.41e-17  1 -0.932 -0.932 -0.932
## 8 child_yes              0             1 -7.69e-17  1 -1.68  -1.68   0.593
##     p75  p100 hist
## 1 0.583 2.89
## 2 1.25  1.25
## 3 0.900 1.77
## 4 0.760 1.59
## 5 0.982 1.53
## 6 1.07  1.07
## 7 1.07  1.07
## 8 0.593 0.593
```

There are three preprocesses applied in this model For step_downsample, it down the sample size affair that really had the bias on “No” answer to be balance on both of yes and no answer, 117:117. For step_dummy, this process converts categorical predictors to the dummy that represent as sex_male and child_yes instead of sex and child columns. Finally, our predictors are also be centered and scaled following data summary and Table6&7.

Tune and fit a model

01 This section is concerned with the tuning and fitting of the model. We will be looking at a k-nearest neighbours model. When considering a k-nearest neighbours model, we need to choose a suitable value for k.

```
affair_knn_spec <- nearest_neighbor( mode = "classification",
                              neighbors = tune()
                              ) %>%
  set_engine("kknn")
# affair_knn <- affair_knn_spec %>%
#   fit(affair ~ ., data = affair_train_preprocess)
# affair_knn
```

02 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)
affair_cv <- vfold_cv(affair_train_preprocess, v = 5)
affair_cv
```

```
## #  5-fold cross-validation
## # A tibble: 5 x 2
##   splits           id
##   <list>           <chr>
## 1 <split [187/47]> Fold1
## 2 <split [187/47]> Fold2
## 3 <split [187/47]> Fold3
## 4 <split [187/47]> Fold4
## 5 <split [188/46]> Fold5
```

03 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.

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

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

```
affair_tune <- tune_grid( affair_knn_spec,
                       preprocessor = recipe(affair ~ ., data = affair_train_preprocess),
                       resamples = affair_cv,
                       grid = affair_grid )
```

05 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_auc <- select_best( affair_tune, "accuracy" )
best_auc
```

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

06 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).

```
affair_knn_spec_final <- finalize_model( affair_knn_spec, best_auc )
affair_knn_spec_final
```

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

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

```
affair_knn <- affair_knn_spec_final %>%
  fit(affair ~ ., data = affair_train_preprocess)
affair_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

We will now evaluate how well our model is at predicting outcomes on new data. This is vital when you have built a model, so that you can have an accurate understanding of how reliable your predictions will be.

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

```
affair_predict <- predict(affair_knn,
                          new_data = affair_test_preprocess,
                          type = "class")

kable(head(affair_predict), caption = "Display the first 6 rows of preprocessed test set using
  kable_styling(latex_options = "hold_position")
```

Table 8: Display the first 6 rows of preprocessed test set using predict

.pred_class
no
no
no
yes
yes
no

02 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.

```
affair_predict <- affair_predict %>%
  bind_cols(affair_test_preprocess %>%
              select(affair))

kable(head(affair_predict), caption = "Display the first 6 rows of preprocessed test set using
  kable_styling(latex_options = "hold_position")
```

03 Get a confusion matrix from your predictions.

```
affair_con_matrix <- affair_predict %>%
  conf_mat(truth = .pred_class, estimate = affair )

affair_con_matrix
```

```
##             Truth
```

Table 9: Display the first 6 rows of preprocessed test set using predict with truth columns

.pred_class	affair
no	no
no	no
no	no
yes	no
yes	no
no	no

```
## Prediction no yes
##        no  81  37
##        yes 11  22
```

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

```
categorical_metrics <- metric_set(sens, spec, precision, recall)
affair_metric <- affair_predict %>%
  categorical_metrics(
    truth = affair,
    estimate = .pred_class
    )

kable(affair_metric, caption = "Evaluation Matrix from predicted test set ") %>%
  kable_styling(latex_options = "hold_position")
```

Table 10: Evaluation Matrix from predicted test set

.metric	.estimator	.estimate
sens	binary	0.6864407
spec	binary	0.6666667
precision	binary	0.8804348
recall	binary	0.6864407

Following the affair confusion matrix calculation, the sensitivity or True positive rate equal to 0.686. It could interpret that there are 68.6% of a positive test results conditioned on individual truly being positive. Therefore, there are 68.644% of those who do not have an affair have been correctly classified as not having an affair. On the other hand, specificity represent the true negative rate. Therefore, it means that there are 66.7% of a negative test results conditioned on individual truly being negative. Finally, there are 66.7% of those who have an affair have been correctly classified as having an affair.

05 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.

```
# Make a tibble containing Bono's information.
new_data = tibble(
  age = 47,
  ym = 15,
  child = "no",
  sex = "male",
  religious = 2,
  occupation = 6,
  education = 20,
  rate = 5
)
# Use bake to preprocess Bono's information with your recipe
new_data_preprocess <- bake( affair_recipe, new_data)

# Using the predict() function, obtain a predicted probability
predict(affair_knn,
        new_data = new_data_preprocess,
        type = "prob")
```

```
## # A tibble: 1 x 2
##   .pred_no .pred_yes
##      <dbl>     <dbl>
## 1    0.375     0.625
```

Whether the result of prediction present that Bono will have an affair or not?, making predictions related to someone's personal life, including sensitive matter, it should be approached with caution. Based on limited information from this situation, it would not be proper to make a prediction and interfere in their personal lives.