Female Affair Classification

2022-08-08

Data Cleaning

```
# Load the necessary libraries.
pacman::p_load(tidyverse,dplyr,lubridate,inspectdf,tidyr,stringr,data.table,caret,tidymodels,skimr,glmx
# Load the data.
data <- read_csv("C:\\Users\\Admin\\Downloads\\affairs.csv")
```

```
## Rows: 601 Columns: 9
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (2): sex, child
## dbl (7): affair, age, ym, religious, education, occupation, rate
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# Read the data as Tibble.
data <- as_tibble(data)
# Display first 6 rows
head(data)
```

```
## # 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
```

Outcome Variable : affair

Predictor Variables : sex, age, ym, child, religious, education, occupation, rate

```
skim_without_charts(data)
```

Table 1: Data summary

Name	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
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

Dataset is having no missing data.

Observations : 601

Variables : 9

Variables affair, sex, and child have been read incorrectly. All three variables should be in factor type.

```
# Convert the affair variable to yes/no response.
data$affair <- ifelse(data$affair == 0, "no","yes")
# Change all character variables to factors.
data$affair <- factor(data$affair)
data$sex <- factor(data$sex)
data$child <- factor(data$child)
```

```
skim_without_charts(data)
```

Table 4: Data summary

Name	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
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

```
count(data,affair)
```

```
## # A tibble: 2 x 2
##   affair     n
##   <fct>  <int>
## 1 no       451
## 2 yes      150
```

```
count(data,child)
```

```
## # A tibble: 2 x 2
##   child     n
##   <fct> <int>
## 1 no      171
## 2 yes     430
```

People having an affair : 150 out of 601

People having children : 430 out of 601

```
mean(data$age)
```

```
## [1] 32.48752
```

```
mean(data$religious)
```

```
## [1] 3.116473
```

Mean age of respondents : 32.487 (Variable type - Quantitative Discrete)

Mean religious scale of respondents : 3.116 (Variable type - Quantitative Discrete)

Exploratory Analysis

```
x <- data %>% filter(affair == "no") %>% count(sex)
x
```

```
## # A tibble: 2 x 2
##   sex        n
##   <fct>  <int>
## 1 female   243
## 2 male     208
```

```
female_prop_no <- 243/451
female_prop_no
```

```
## [1] 0.5388027
```

```
x <- data %>% filter(affair == "yes") %>% count(sex)
x
```

```
## # A tibble: 2 x 2
##   sex        n
##   <fct>  <int>
## 1 female    72
## 2 male      78
```

```
female_prop_yes <- 72/150
female_prop_yes
```

```
## [1] 0.48
```

Proportion of female who responded “no” to an affair : 0.539

Proportion of female who responded “yes” to an affair : 0.48

There does appear to be a diiference. The proportion of females who will not have an affair is greater than the proportion of females who will have an affair.

```r
x <- data %>% filter(affair == "yes") %>% count(child)
x
```

```
## # A tibble: 2 x 2
##   child     n
##   <fct> <int>
## 1 no       27
## 2 yes     123
```

```r
child_prop_yes <- 123/150
child_prop_yes
```

```
## [1] 0.82
```

```r
x <- data %>% filter(affair == "no") %>% count(child)
x
```

```
## # A tibble: 2 x 2
##   child     n
##   <fct> <int>
## 1 no      144
## 2 yes     307
```

```r
child_prop_no <- 307/451
child_prop_no
```

```
## [1] 0.6807095
```

Proportion of participants who responded "yes" to having an affair had children : 0.82

Proportion of participants who responded "no" to having an affair had children : 0.681

You are more likely to have children if you have an affair.

Split and Preprocess

```r
# Set the seed to handle the randomness.
set.seed(2022)
# Split the data with 75-25% ratio relative to training and testing set.
split <- initial_split(data)
split
```

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

Observations in training set : 450 out of 601

Observations in testing set : 151 out of 601

```
# Obtain training set named "train".
train <- training(split)
# Obtain testing set named "test".
test <- testing(split)
# Display first 6 observations of training set.
head(train)
```

```
## # 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      27   1.5 yes           2        16          2     4
## 2 no     female    27   7   yes           2        12          1     2
## 3 yes    female    42  15   yes           4        16          5     4
## 4 no     male      42  15   yes           4        17          5     3
## 5 no     female    42  15   yes           4        18          6     5
## 6 no     female    27   4   yes           3        18          4     5
```

step_downsample package removes the biasness. This package ensures/equals the occurance/frequency of each lables in the outcome variable.

The reason behind down sample this data is that, traning set contians of 116 out of 450 obvervations as "yes" to an affair, and 334 out of 450 observations as "no" to an affair. Therefore, the proportion of observation having an affair is lower than the not having an affair. That causes biasness. Hence, down sample the data is necessary.

```
# Create a recipe based on training set.
ad_recipe <- recipe( affair ~ ., data = train ) %>%
  # Down sample the data
  themis::step_downsample(affair) %>%
  # Convert both categorical predictors to dummy variables.
  step_dummy(all_nominal(),all_predictors(),-all_outcomes()) %>%

  # Optional method for dummy variables.
    # step_dummy( sex ) %>%
    # step_dummy( child ) %>%

  # Normalize all predictor variables with mean 0 and std_dev 1.
  step_normalize( all_predictors() ) %>%
  prep()
```

```
## Warning: The following variables are not factor vectors and will be ignored:
## `age`, `ym`, `religious`, `education`, `occupation`, `rate`
```

```
ad_recipe
```

```
## Recipe
##
## Inputs:
##
##       role #variables
##    outcome          1
```

```
##  predictor          8
##
## Training data contained 450 data points and no missing data.
##
## Operations:
##
## Down-sampling based on affair [trained]
## Dummy variables from sex, child [trained]
## Centering and scaling for age, ym, religious, education, occupation, rate... [trained]
```

```
# Obtain preprocessed training set named "pre_train".
ad_juiced <- juice( ad_recipe )
pre_train <- bake( ad_recipe, new_data = NULL)
# Obtain preprocessed testing set named "pre_test".
pre_test <- bake( ad_recipe, new_data = test)
```

```
pre_train %>%
  skim_without_charts()
```

Table 7: Data summary

Name	Piped data
Number of rows	232
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: 116, yes: 116

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
age	0	1	0	1	-1.68	-0.61	-0.05	0.52	2.77
ym	0	1	0	1	-1.51	-0.79	-0.24	1.25	1.25
religious	0	1	0	1	-1.68	-0.83	0.01	0.85	1.69
education	0	1	0	1	-2.99	-0.89	-0.05	0.79	1.63
occupation	0	1	0	1	-1.85	-0.71	0.44	0.58	1.58
rate	0	1	0	1	-2.44	-0.68	0.20	1.08	1.08
sex_male	0	1	0	1	-1.04	-1.04	0.96	0.96	0.96
child_yes	0	1	0	1	-1.75	0.57	0.57	0.57	0.57

Traing set("train") contains 450 observations. However, by observing the above chart, after the down sample the training set, now trining set("pre_train") is containing 232 observations with equal proportions of both "yes" and "no" regarding the outcome variable(affair).

Both “sex” and “child” variables are converted into dummy variables with “sex_male” and “child_yes” variables as numeric type.

By observing the above chat, you can see the all numerical variables are having mean 0 and sd 1.

Hence, our all 3 preprocessing steps have done successfully.

Tune and Fit a Model

```
# Create a model named "knn" containing model specification.
# For optiomal value of "k", set the neighbors parameter to tune().
knn <- nearest_neighbor( mode = "classification", neighbors = tune() ) %>%
  set_engine("kknn")
knn
```

```
## K-Nearest Neighbor Model Specification (classification)
##
## Main Arguments:
##   neighbors = tune()
##
## Computational 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(1223).

```
# Set the seed to handle the randomness.
set.seed(1223)
# Create a 5-fold cross-validation set from preprocessed training set named "cv".
cv <- vfold_cv(pre_train, 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.

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

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

```
# Create a tuned model named "knn_tuned".
knn_tuned <- tune_grid(object = knn,
                       preprocessor = recipe(affair ~ ., data =  pre_train),
                       resamples = cv,
                       grid = knn_grid)
```

```
## Warning: package 'kknn' was built under R version 4.2.1
```

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

```
## # A tibble: 1 x 2
##   neighbors .config
##       <int> <chr>
## 1         7 Preprocessor1_Model02
```

The optimal value of k : 7

```
# Create a finalized model named "knn_model" with optimal value for "k".
knn_model <- finalize_model(knn, best_acc)
knn_model
```

```
## K-Nearest Neighbor Model Specification (classification)
##
## Main Arguments:
##   neighbors = 7
##
## Computational engine: kknn
```

```
affairs_knn <- knn_model %>%  fit(affair ~ ., data=pre_train)
affairs_knn
```

```
## parsnip model object
##
##
## Call:
## kknn::train.kknn(formula = affair ~ ., data = data, ks = min_rows(7L,     data, 5))
##
## Type of response variable: nominal
## Minimal misclassification: 0.3534483
## Best kernel: optimal
## Best k: 7
```

Evaluation

```
# Predict the prepossessed testing set.
preds <- affairs_knn %>%
  predict(new_data = pre_test,
          type = "class")
# Display the first six observations.
head(preds)
```

```
## # A tibble: 6 x 1
##   .pred_class
##   <fct>
## 1 no
## 2 yes
```

```
## 3 yes
## 4 no
## 5 yes
## 6 no
```

```
preds_affair <- preds %>%
  bind_cols(pre_test %>%
              dplyr::select(affair))
```

```
preds_affair %>%
  conf_mat(truth = affair, estimate = .pred_class)
```

```
##           Truth
## Prediction no yes
##        no  56  16
##        yes 61  18
```

```
specificity(preds_affair, truth = affair, estimate = .pred_class) %>%
  bind_rows(sensitivity(preds_affair, truth = affair, estimate = .pred_class))
```

```
## # A tibble: 2 x 3
##   .metric     .estimator .estimate
##   <chr>       <chr>          <dbl>
## 1 specificity binary         0.529
## 2 sensitivity binary         0.479
```

A model with high specificity means that model is correctly predicting most of the negative results("no"). In this case model is not performing well with identifying the people not having an affair.

A model with high sensitivity means that model is correctly predicting most of the positive results("yes"). In this case model is not performing well with identifying the people having an affair.

. 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.

```
newdata <- tibble(
  sex = factor("male"),
  age = 47,
  ym = 15,
  child = factor("no"),
  religious = 2,
  education = 20,
  occupation = 6,
  rate = 5)
```

```
newdata <- bake( ad_recipe, new_data = newdata)
```

```
new_pred <- affairs_knn %>%
  predict(new_data = newdata,
          type = "prob")
```

```
new_pred
```

```
## # A tibble: 1 x 2
##   .pred_no .pred_yes
##      <dbl>     <dbl>
## 1    0.163     0.837
```

With the 0.837 probability, We can say that, Bono will have an affair.