MATHS 7107 Data Taming Assignment Five

Chang Dong

2023-04-11

Due date: 5pm, Wednesday 12th April 2023.

Some rules about your submissions:

- **You must complete this assignment using R Markdown;**
- Your assignment must be submitted as **pdf only** on MyUni;
- You must include **units** when providing solutions;
- Include any working when providing solutions;
- Provide all numerical answers to **3 decimal places**;
- Make sure you include both your code and R output / plots in your answers;
- Make sure any tables or plots included have captions;
- Do not write directly on the question sheet;
- You can submit more than once if you find errors and your latest submission will be marked;
- Make sure you only upload one document for your final submission. If you submit multiple pages (i.e. one per question) you will be deducted 10% per page submitted;
- Penalties for late submission - within 24 hours 40% of final mark. After 24 hours, assignment is not marked and you get zero; and
- Finally, make sure you check your submitted assignment is the correct one, as we cannot accept other submissions after the due date.

The purpose of this assessment is to work through all the relevant steps of constructing a predictive model on a given data set. This will include data cleaning, exploratory analysis, preprocessing, model building and tuning, and finally model evaluation and predicting on new data. This will assess a variety of skills developed in the course.

Label all question answers clearly. Include any relevant code in your answers, as there are marks awarded for code.

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.

Further information regarding this data can be found in the original paper:

Fair, R. C. (1978). A theory of extramarital affairs. Journal of Political Economy, 86(1), 45-61.

This includes a more in depth discussion of the data and variables involved.

Loda pkgs

```
pacman::p_load(skimr, tidyverse, tidymodels, themis, recipes, dials,kknn)
```

Data Cleaning

The questions in this section will involve making sure the data is all good to work with. Data cleaning processes include checking for missing data, making sure variables have been read in correctly, and making sure the values of our data are reasonable.

1. 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 (head is a good function for this). [2]

```
affairs <- as_tibble(read.csv("affairs.csv"))
head(affairs)
```

```
## # A tibble: 6 x 9
##   affair sex      age    ym child religious education occupation  rate
##    <int> <chr>  <dbl> <dbl> <chr>     <int>     <int>      <int> <int>
## 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
```

2. What is the outcome variable, and what are the predictor variables? [2]

The outcome variable is "affair", and the rest of variables are predictors("sex", "age", "ym", "child", "religous", "education", "occupation", "rate").

3. Skim the data. Is there any missing data? How many observations and variables do we have? Have any variables been read in incorrectly? [4]

```
skim(affairs)
```

Table 1: Data summary

Name	affairs
Number of rows	601
Number of columns	9
Column type frequency:	
character	2

Table 1: Data summary

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	hist
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	

missing data: There is no missing data in our tibble.

observations and variables: It totally have 601 observations and 9 variables.

variables incorrectly: "sex" and "child" have been read incorrectly, because they are categorical variable, should be factor, but they were character in the table. "affair" was also read incorrectly into the numerical(int), it has 2 categories, 1 and 0, should be factor.

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. [3]

```
affairs_tb <- affairs %>%
  mutate( sex= factor(sex),
          child = factor(child),
          affair = factor( ifelse(affair == 1, "yes", "no"))
   ) %>%
  mutate(affair = factor(affair))

skim(affairs_tb)
```

Table 4: Data summary

Name	affairs_tb
Number of rows	601
Number of columns	9
Column type frequency:	
factor	3
numeric	6

Table 4: Data summary

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	hist
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	

5. Skim the data again and answer the following. [5]

a. How many people responded as having had an affair? How many people responded to having children?

From the result of `skim_without_charts(affairs_tb)` in 4, we know that **150** people responded to having had an affair, **430** people responded to having children.

b. What is the mean age of respondents? What is the mean response on the religious scale?

From the result of `skim_without_charts(affairs_tb)` in 4, we know that the mean age of respondents is **32.488**, and the mean response on the religious scale is **3.116**.

Section Total: [16]

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 visualisation.

1. 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) [3]

```
affair_no <- affairs_tb %>%
  filter(affair == "no") %>%
  count(sex)

affair_no_prop <- round(affair_no[affair_no$sex == "female",]$n / sum(affair_no$n),3)
affair_no_prop
```

```
## [1] 0.539
```

```
affair_yes <- affairs_tb %>%
  filter(affair == "yes") %>%
  count(sex)

affair_yes_prop <- round(affair_yes[affair_yes$sex == "female",]$n / sum(affair_yes$n),3)
affair_yes_prop
```

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

Of the participants who responded "no" to an affair, the proportion of them are female is **0.539**.

Of the participants who responded "yes" to an affair, the proportion of them are female is **0.480**.

It shows **a small difference** in the proportion of females who will have an affair opposed to those who will not.

2. 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? [3]

```
affair_no_child <- affairs_tb %>%
  filter(affair == "no") %>%
  count(child)


affair_no_child_prop <- round(affair_no_child[affair_no_child$child == "yes",]$n
                              / sum(affair_no_child$n),3)
affair_no_child_prop
```

```
## [1] 0.681
```

```
affair_yes_child <- affairs_tb %>%
  filter(affair == "yes") %>%
  count(child)


affair_yes_child_prop <- round(affair_yes_child[affair_yes_child$child == "yes",]$n
                               / sum(affair_yes_child$n),3)
affair_yes_child_prop
```

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

The proportion of participants who responded "yes" to having an affair had children is **0.820**.

The proportion of participants who responded "no" to having an affair had children is **0.681**.

Based on this, people **more likely** to have children if they have an affair. (**0.820 > 0.681**)

Usually we would look at plots to compare the relationships in our dataset using summary statistics and data visualisation. To reduce the length of the assignment, we won't do that this week

Section Total: [6]

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.

1. 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). [3]

```
set.seed(1234)
affairs_split <- initial_split( affairs_tb, strata = affair )
affairs_split
```

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

450 observations are in the training set and **151** observations are in the testing set.

2. 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. [3]

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

```
## # A tibble: 6 x 9
##   affair sex      age    ym child religious education occupation  rate
##   <fct>  <fct>  <dbl> <dbl> <fct>     <int>     <int>      <int> <int>
## 1 no     female    27   4   no            4        14          6     4
## 2 no     male      57  15   yes           5        18          6     5
## 3 no     female    32   1.5 no            2        17          5     5
## 4 no     male      22   1.5 no            4        14          4     5
## 5 no     male      27   4   yes           4        18          6     4
## 6 no     male      47  15   yes           5        17          6     4
```

```
affairs_train %>% count(affair) %>%
  mutate(percent = prop.table(n) * 100)
```

```
## # A tibble: 2 x 3
##   affair     n percent
##   <fct>  <int>   <dbl>
## 1 no       338    75.1
## 2 yes      112    24.9
```

```
affairs_test %>% count(affair)  %>%
  mutate(percent = prop.table(n) * 100)
```

```
## # A tibble: 2 x 3
##   affair     n percent
##   <fct>  <int>   <dbl>
## 1 no       113    74.8
## 2 yes       38    25.2
```

The stratified split makes sure that the percentage of each category of affair shows a similar proportion both in the training and testing set.

3. What does step_downsample from the themis package do? Why might we want to down sample our data? [4]

The `step_downsample` was used to tackle the imbalanced problem of a data set. it can remove some samples to keep all classes in the same amount level. When there is a majority class that greatly exceeds the other class(es), it may influence the training performance. So we need to use `step_downsample` to remove some

samples in the majority classes that ensure the count of all classes keeps at a similar level. Thus, it improves the performance when training a model using the imbalanced data set.

4. In tutorial 3 we saw how to use recipes. Create a recipe, based off of our training data, that will: [4]

- Down sample our data on affair. Do this using themis::step_downsample(affair) in your recipe,

```
affair_rcp <- recipe(affair ~ ., data = affairs_train) %>%
  step_downsample(affair)
```

- Convert all our categorical predictors to dummy variables, and

```
affair_rcp <- affair_rcp %>% step_dummy(all_nominal(), -all_outcomes())
```

- Normalise all of our predictor variables to have mean 0 and standard deviation 1.

```
affair_rcp <- affair_rcp %>% step_normalize(all_predictors())
```

- Print out the recipe to make sure it has worked using prep().

```
affair_rcp <- affair_rcp %>% prep()
affair_rcp
```

```
##
## -- 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. [1]

```
affair_train_prep <- juice(affair_rcp)
```

```
b. Use the function bake (on the recipe and testing split) to get your preprocessed testing set. This ca
```

```
affair_test_prep <- bake(affair_rcp, new_data = affairs_test)
```

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

```
skim(affair_train_prep)
```

Table 7: Data summary

Name	affair_train_prep
Number of rows	224
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: 112, yes: 112

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
age	0	1	0	1	-1.66	-0.63	-0.10	0.44	2.59	
ym	0	1	0	1	-1.57	-0.85	-0.30	1.19	1.19	
religious	0	1	0	1	-1.64	-0.84	-0.03	0.77	1.58	
education	0	1	0	1	-2.96	-0.90	0.13	0.75	1.58	
occupation	0	1	0	1	-1.93	-0.79	0.36	0.93	1.51	
rate	0	1	0	1	-2.37	-0.65	0.22	1.08	1.08	
sex_male	0	1	0	1	-1.01	-1.01	0.99	0.99	0.99	
child_yes	0	1	0	1	-1.73	0.00	0.58	0.58	0.58	

1. the numebr of observations reduced to 224 because some "no" samples were removed from the training set, and now the affair was class balanced (112 of yes, and 112 of no).
2. There are no factors in our predictors, all the categorical variables were converted to dummy variables(one-hot variables).
3. All predictors were normalized, and have the mean of around 0, and the standard deviation of 1.

Section Total: [20]

Tune and fit a model

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.

1. Make a model specification for a k-nearest neighbours model. In the model specification, define that we would like to tune() the neigbors parameter. [2]

```
near_neighbour_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). [3]

```
set.seed(1234)
affair_cv <- vfold_cv(
 data = affair_train_prep,
 v = 5,
 strata = affair)

affair_cv  %>%
  slice( 1 ) %>%
 pull( splits )
```

```
## [[1]]
## <Analysis/Assess/Total>
## <178/46/224>
```

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. [2]

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

k_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
## # ... with 15 more rows
```

4. Use tune_grid to tune your k-nearest neighbours model using your cross validation sets and grid of k-values. [2]

```
affair_knn_tune <- tune_grid(
  object = near_neighbour_spec,
  resamples = affair_cv,
  grid = k_grid,
  preprocessor = recipe(affair ~., data = affair_train_prep),
)
```

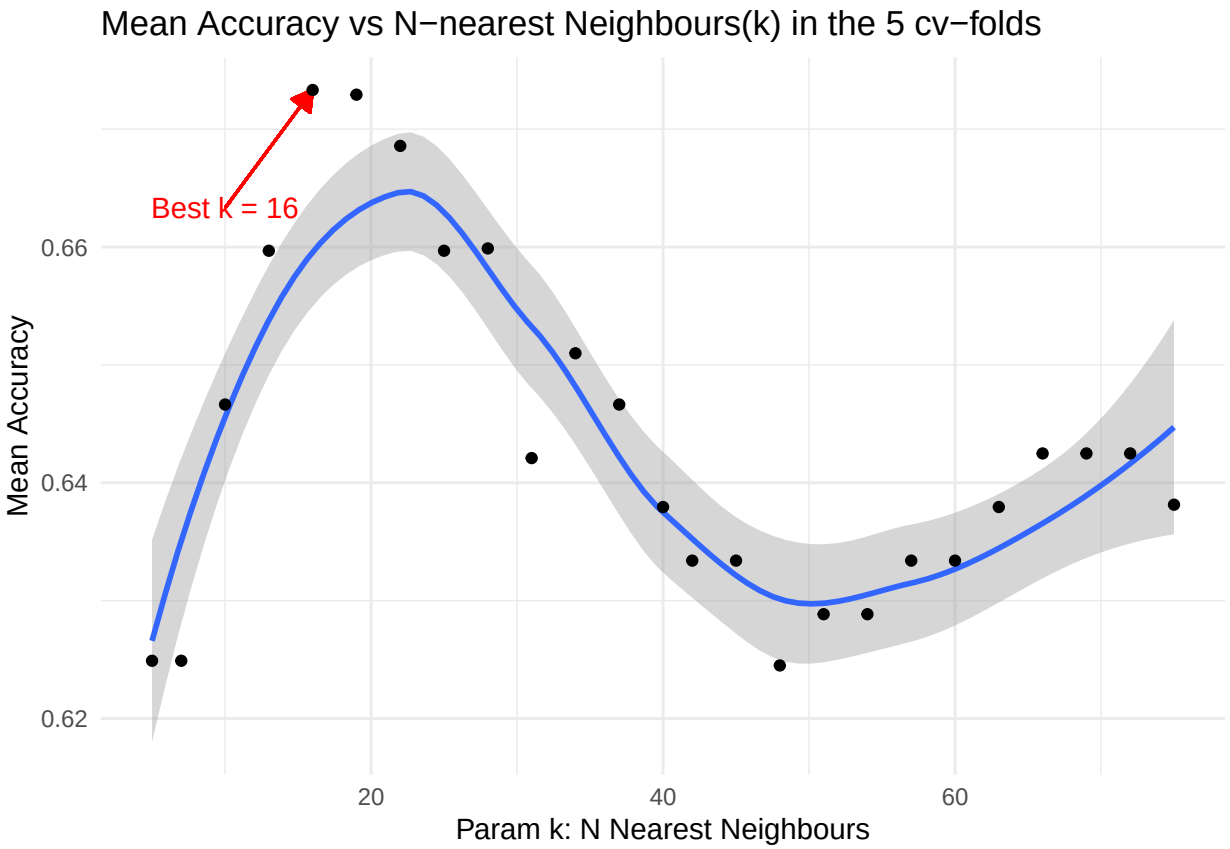
5. What is the value of k that gives the best accuracy based on our tuned model? (Hint: the function select_best will be useful with tuned model as the first parameter and "accuracy" as the second parameter) [2]

```
best_k <- select_best(affair_knn_tune, "accuracy")
best_k
```

```
## # A tibble: 1 x 2
##   neighbors .config
##       <int> <chr>
## 1        16 Preprocessor1_Model05
```

```
cv1<-as.data.frame(affair_knn_tune$.metrics[1])[1:25,4]
cv2<-as.data.frame(affair_knn_tune$.metrics[2])[1:25,4]
cv3<-as.data.frame(affair_knn_tune$.metrics[3])[1:25,4]
cv4<-as.data.frame(affair_knn_tune$.metrics[4])[1:25,4]
cv5<-as.data.frame(affair_knn_tune$.metrics[5])[1:25,4]
k <- as.data.frame(affair_knn_tune$.metrics[1])[1:25,1]
accuracy <- (cv1+cv2+cv3+cv4+cv5)/5
acc_df <- data.frame(k = k, accuracy = accuracy)
best_acc <- acc_df %>% filter(k == 16) %>% select(accuracy)


acc_df %>% ggplot(aes(x = k, y = accuracy)) +
geom_smooth(method ="loess",formula = 'y ~ x') +
ggtitle("Mean Accuracy vs N-nearest Neighbours(k) in the 5 cv-folds")+
xlab("Param k: N Nearest Neighbours") +
ylab("Mean Accuracy") + theme_minimal() +
geom_segment(aes(x = 10, y = best_acc$accuracy - 0.01,
                 xend = 16, yend = best_acc$accuracy),
                 arrow = arrow(length = unit(0.3, "cm"),
                               type = "closed"),
                 color = "red") +
annotate("text", x = 10, y = best_acc$accuracy - 0.01,
         label = "Best k = 16", size = 4, color = "red") +
geom_point()
```



From the 'select_best(affair_knn_tune, "accuracy")' code and the figure, we knows the best k occurs at k = 16, because when k = 16, the mean accuracy is the highest.

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) [2]

```
affair_knn_final <- nearest_neighbor( mode = "classification", neighbors = 16 ) %>%
  set_engine( "kknn" ) %>% finalize_model(select_best(affair_knn_tune,"accuracy"))
affair_knn_final
```

```
## K-Nearest Neighbor Model Specification (classification)
##
## Main Arguments:
##   neighbors = 16
##
## Computational engine: kknn
```

7. Fit your finalised model to the preprocessed training data and save it with the variable name affairs_knn. [1]

```
affairs_knn <- affair_knn_final %>%
  fit( affair ~ ., data = affair_train_prep )
affairs_knn
```

```
## parsnip model object
##
##
## Call:
## kknn::train.kknn(formula = affair ~ ., data = data, ks = min_rows(16,     data, 5))
```

```
##
## Type of response variable: nominal
## Minimal misclassification: 0.3348214
## Best kernel: optimal
## Best k: 16
```

Section Total: [14]

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.

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. [2]

```
affair_knn_pred <- predict(affairs_knn,
                           new_data = affair_test_prep)
head(affair_knn_pred)
```

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

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. [2]

```
affair_knn_prediction <- affair_knn_pred %>%
  bind_cols( select(affair_test_prep, affair) )
colnames(affair_knn_prediction) = c("predicted", "true_label")
head(affair_knn_prediction)
```

```
## # A tibble: 6 x 2
##   predicted true_label
##   <fct>     <fct>
## 1 no        no
## 2 yes       no
## 3 no        no
## 4 no        no
## 5 no        no
## 6 yes       no
```

3. Get a confusion matrix from your predictions. [2]

```
cfm <- table(affair_knn_prediction)
cfm
```

```
##          true_label
## predicted no yes
##       no  66  17
##       yes 47  21
```

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

```
Sensitivity <- round(cfm[2,2]/(cfm[2,2]+cfm[1,2]),3) #round(21/(17+21),3)
Specificity <- round(cfm[1,1]/(cfm[1,1]+cfm[2,1]),3) #round(66/(66+47),3)
Sensitivity
```

```
## [1] 0.553
```

```
Specificity
```

```
## [1] 0.584
```

In this case, we define the positive is "yes", and negative is "no"

Sensitivity = (True Positive)/(True Positive + False Negative) means the amount of "affair = yes" that we can predicted account for all the real "affair = yes".which is 0.553.

Specificity = (True Negative)/(True Negative + False Positive) means the amount of "affair = no" that we can predicted account for all the real "affair = no". which is 0.584.

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. Be careful when you do this. You need to name your variables exactly how they appear in the affairs dataset (i.e. sex, age, etc). Remember, case matters. [1]

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

```
## # 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 your recipe. [2]

```
newdata_bake <- bake(affair_rcp, new_data = newdata )
newdata_bake
```

```
## # A tibble: 1 x 8
##     age    ym religious education occupation  rate sex_male child_yes
##   <dbl> <dbl>     <dbl>     <dbl>      <dbl> <dbl>    <dbl>     <dbl>
## 1  1.52  1.19    -0.837      1.58      0.934  1.08    0.989     -1.73
```

c. Using the predict() function, obtain a predicted probability (i.e. with type = "prob") that Bono will have an affair. [2]

```
newdata_pred <- predict(affairs_knn,
                        new_data = newdata_bake,
                       type = "prob") %>% round(3)
newdata_pred
```

```
## # A tibble: 1 x 2
##   .pred_no .pred_yes
```

```
##       <dbl>      <dbl>
## 1      0.31       0.69
```

d. Given what we have done, would you be comfortable going to Bono's partner with your prediction of whether Bono will have an affair or not? [2]

No. Firstly and objectively, result of this model didn't gives a very high accuracy, so it is not reliable. Secondly, for rthical considerations, this prediction may influence the trust of the couple. Finaly, even the prediction give a very high probability, different individual may shows big variance.

Section Total: [**17**]

Assessment Total: [**73**]