Assignment_5

aditya venugopalan a1899824

2023-11-08

Data Cleaning

1 Importing Dataset

```
library(readr)
affair <- read_csv("affairs.csv", show_col_types = FALSE)
```

```
affair <- as_tibble(affair)
head(affair)
```

```
## # 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
```

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

#Outcome variable :- “affair”. Since the entire predictive model will be built to determine whether an individual will be involved in an affair #Predictor variables :- sex, age, ym, child, religious,education, occupation and 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?

```
affair %>%
  skim_without_charts()
```

Table 1: Data summary

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
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

Observations

#From the above results we can conclude the following: # a) No there is missing data # b) There are 601 observations accompanied by 9 variables . # c) It was observed that the variables weren't read properly as the variable "affair" which was supposed to be a categorical variable but was read as numerical variable. Also sex and child were read as characters, instead of being read as factors.

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.

```r
affair <- affair %>%
  mutate(affair= case_when(affair == 1 ~ "yes",
                           TRUE ~ "no")) %>%
  mutate_if(is.character, factor)
```

5. Skim the data again and answer the following.

```r
skim_without_charts(affair)
```

Table 4: Data summary

Name	affair
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) There are 150 people in total who have responded to "yes" and 451 have responded "no" to having an affair. In total 430 people have declared having children and 171 people have said they don't children.

b) The mean age of respondents is found out to be 32.488.

#The mean response for religious is 3.116

Exploratory analysis

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)

```
affair %>%
  count(affair, sex) %>%
  pivot_wider(names_from = sex, values_from = n) %>%
  mutate(proportion = female/ (male+female))
```

```
## # A tibble: 2 x 4
##   affair female  male proportion
##   <fct>   <int> <int>      <dbl>
## 1 no        243   208      0.539
## 2 yes        72    78      0.48
```

fFrom the above observations we can say females who have confessed to have an affair and the females who have confess to having not an affair are 53% and 48% respectively.

From the above observations we can see that the proportion of females who haven't done an affair are comparatively more than the proportion of females who had an affair

```
affair %>%
  count(affair, child) %>%
  pivot_wider(names_from = child, values_from = n) %>%
  mutate(proportion = yes/(yes + no))
```

```
## # A tibble: 2 x 4
##   affair    no   yes proportion
##   <fct>  <int> <int>      <dbl>
## 1 no       144   307      0.681
## 2 yes       27   123      0.82
```

From the above observations we can say that female who are doing affair with childeren is 82% and those who are not involved in affair but have a child are 68%.From the above observations we can conclude that , people having children are more favourites to engage in an affair .

Split and preprocess

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)

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

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

#From using the initial_split we created rsplit and by doing that we can observe that there are 601 total number of observations , 450 observations in training set and 151 observations 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.

```
affair_training <- training(affair_split)
affair_test <- testing(affair_split)
head(affair_training)
```

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

The step_downsample is used for randomly removing observations from the majority data category in our dataset which in our case is "no" to affairs. Down sampling is used when there are highly unbalanced groups, by using down sampling we can make sure there is equality in different categories in our model. Few other advvantages of step_downsample includes , reducing the the size of dataset while not losing the characteristics, faster training of the model and reducing bias .

4 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_training) %>%
  themis::step_downsample(affair) %>%
  step_dummy(all_nominal(), -all_outcomes() ) %>%
  step_normalize(all_predictors() ) %>%
  prep()

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
```

5 Complete the following:

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

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. [1]

```
affair_preprocess_trained <- juice(affair_recipe)
affair_preprocess_tested <- bake( affair_recipe, affair_test)
```

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

```
affair_preprocess_trained %>%
  skim_without_charts()
```

Table 7: Data summary

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
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

After the process of Down sampling we can say that , the downsampling was a success as it is evident that the number of observations for the variable affair is equal for the both the categories . The number of rows have reduced to 234 . The categorical variables ex and child are now dummy variables and most importantly every predictor variables are now normalized to 0 and standard deviation is 1 as asked .

Fit Logistic regression

```
lr_affair <- logistic_reg(mode = "classification") %>% set_engine("glm")
 lr_affair_model<- lr_affair %>% fit(affair ~ ., data = affair_preprocess_trained)
```

The motivation or the reason behind using this split is to observe and understand how the model genralizes to new data and hence make necessary evaluation. This also promotes unbiased evaluation of the model. We cannot use the entire dataset for training the model since we have to assess the performance of the model on new data. If we happen to use same data for both testing and training then we are very likely to encounter inaccurate assessment of the model's ability to make prediction on new data. That is why data split is done so that we can evaluate how ell the model will perfom wehn faced with new data point, which will make us sure we are not in a situation whhere there is a possibility of overfitting.

Tune and fit a k-nearest neighbours model

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

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

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

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)
crossvali_affair <- vfold_cv(affair_preprocess_trained, v = 5)
crossvali_affair
```

```
## #  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
```

#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_kgrid <- grid_regular(neighbors(range = c(5,75)),levels = 25)
head(affair_kgrid)
```

```
## # A tibble: 6 x 1
##   neighbors
##       <int>
## 1         5
## 2         7
## 3        10
## 4        13
## 5        16
## 6        19
```

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

```
affair_tune <- tune_grid(object = affair_knn,
                         preprocessor = recipe(affair ~ . , data = affair_preprocess_trained),
                         resamples = crossvali_affair, grid = affair_kgrid )
```

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)

```
affair_bestacc<- select_best(affair_tune, "accuracy")
affair_bestacc
```

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

#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_final <- finalize_model(affair_knn, affair_bestacc)
affair_knn_final
```

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

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

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

Obtain class predictions using both of your finalised models from the preprocessed test set usingpredict. Print the first 6 rows to make sure it worked.

```
predict_lr_model <- predict(lr_affair_model, new_data = affair_preprocess_tested, type = "class")
head(predict_lr_model)
```

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

```
predict_knn_model <- predict(affairs_knn, new_data= affair_preprocess_tested, type = "class")
head(predict_knn_model)
```

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

#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

```
predict_lr_model <- predict_lr_model %>%
  bind_cols(affair_preprocess_tested %>%
              select(affair) ) %>% rename(value_predicted = .pred_class, true_value = affair)
head(predict_lr_model)
```

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

```
predict_knn_model <- predict_knn_model %>%
  bind_cols(affair_preprocess_tested %>%
              select(affair) ) %>% rename(value_predicted = .pred_class, true_value = affair)
head(predict_knn_model)
```

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

Get a confusion matrix from your predictions

```
predict_lr_model %>%
  conf_mat(truth = true_value, estimate = value_predicted)
```

```
##           Truth
## Prediction no yes
##        no  81  16
##        yes 37  17
```

```
sensitivity_lr<- 81/(81+37)
specificity_lr <- 17/(17+16)
```

```
print(sensitivity_lr)
```

```
## [1] 0.6864407
```

```
print(specificity_lr)
```

```
## [1] 0.5151515
```

```
predict_knn_model %>%
  conf_mat(truth = true_value, estimate = value_predicted)
```

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

```
sensitivity_knn<- 81/(81+37)
specificity_knn <- 22/(22+11)
```

```
print(sensitivity_knn)
```

```
## [1] 0.6864407
```

```
print(specificity_knn)
```

```
## [1] 0.6666667
```

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

```
lr_specificity <- predict_lr_model %>%
  spec( truth = true_value, estimate = value_predicted )


lr_sensitivity <- predict_lr_model %>%
  sens( truth = true_value, estimate = value_predicted )


paste( "The sensitivity of the logistic regression model is ", round(lr_sensitivity[,3], 3), "And the t
```

```
## [1] "The sensitivity of the logistic regression model is  0.686 And the the Specificity for the logis
```

```
knn_specificity <- predict_knn_model %>%
  spec( truth = true_value, estimate = value_predicted )


knn_sensitivity <- predict_knn_model %>%
  sens( truth = true_value, estimate = value_predicted )


paste( "The sensitivity of the k-nearest neighbours model is ", round(knn_sensitivity[,3], 3), "And the
```

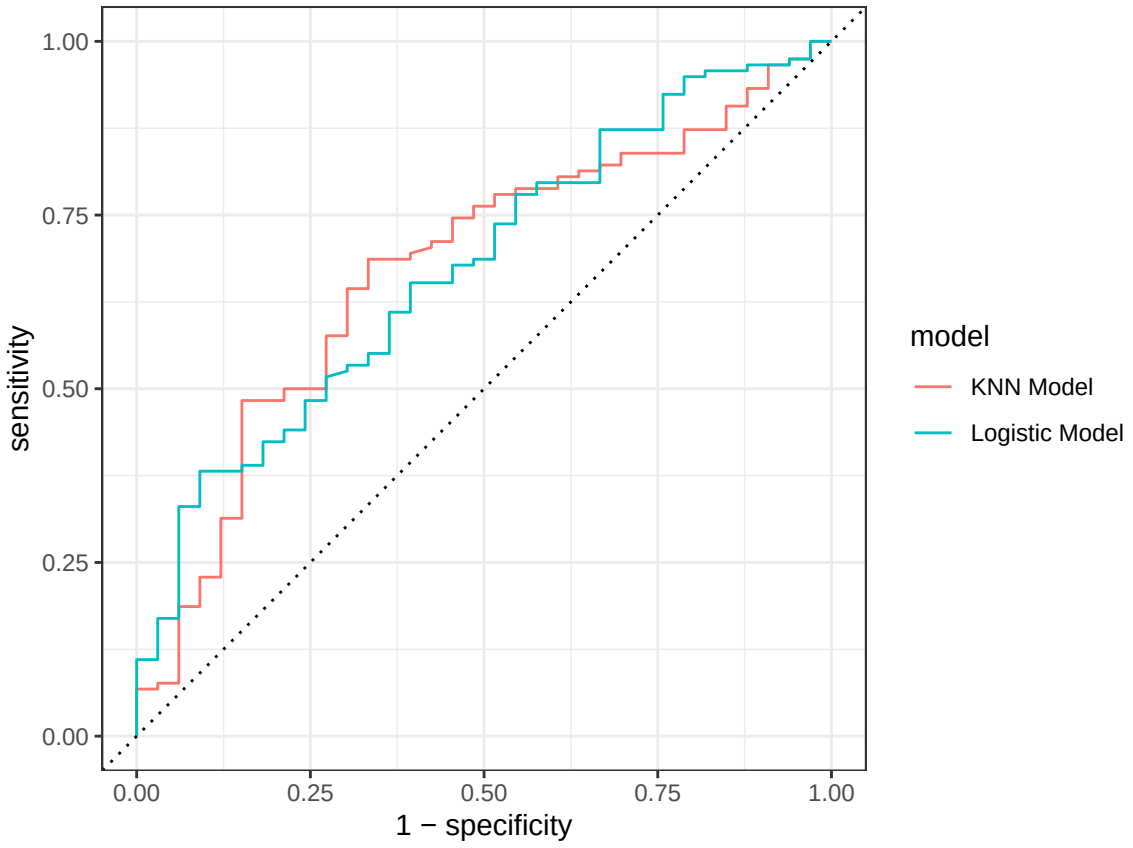
```
## [1] "The sensitivity of the k-nearest neighbours model is  0.686 And the the Specificity for the k-ne
```

In our case , here sensitivity simply means the capacity of the predicton model to correctly tell us the people who have respondend "no" and the specificity tells us about the people who have responded as "yes" to having an affair.

create an ROC curve for both models on the same plot. Which model should be chosen and why?

```
lr_curve_roc <- predict( lr_affair_model,
                new_data = affair_preprocess_tested,
                type = "prob" )  %>%
                  bind_cols( affair_preprocess_tested %>%
                    select( affair ) ) %>%
                      rename(predicted_value = .pred_no, true_value = affair) %>%
                        mutate( model = "Logistic Model") %>%
            bind_rows(
              predict( affairs_knn,
                new_data = affair_preprocess_tested,
                type = "prob" )  %>%
                  bind_cols( affair_preprocess_tested %>%
                    select( affair ) ) %>%
                      rename(predicted_value = .pred_no, true_value = affair) %>%
                         mutate( model = "KNN Model")
            )

lr_curve_roc %>%
  group_by( model ) %>%
    roc_curve( truth = true_value, predicted_value ) %>%
      autoplot()
```




```
lr_curve_roc %>%
  group_by( model ) %>%
    roc_auc( truth = true_value, predicted_value )
```

```
## # A tibble: 2 x 4
##   model          .metric .estimator .estimate
##   <chr>          <chr>   <chr>          <dbl>
## 1 KNN Model      roc_auc binary         0.672
## 2 Logistic Model roc_auc binary         0.673
```

From the above table we can conclude that the Logistic model is the better model and it should be chosen for the following reasons:

a) the logistic regregession model seems to have higher true positive rate , which allows us to safely assumse it is a better model .

b) While one can argue that other factors also play an important role , but since we have the knowledge that genrally the curve that is closer to the top left corner plot is chosen as the better model , we will finish our search of best models by sticking to these two reasons