Sample Answer Lab01: Basic tidyverse operations, simple modeling approach and data engineering

Handed out: Thursday, February 25, 2021

Return date: Thursday, March 11, 2021 by midnight in the *ELEARNING* assignment folder **`Lab01Submit`**.

Grades: Lab01 counts 16 % towards your final grade.

Objectives: Basic tidyverse operations, simple modeling approach and data engineering.

Task 1: Based on Chapter 5 *Data Transformation with dplyr* answer the following questions and show your code. Make sure to attach the libraries **`tidyverse`** and **`nycflights13`** to your session. [5 points]

[a] What is the difference between **==** and **`near( )`**? Give examples using numbers derived from floating point operations. [0.25 points]

`near()` is a safe way of comparing if two vectors of floating-point numbers are (pairwise) equal. This is safer than using ==, because it has a built-in tolerance

```
sqrt(2) ^ 2 == 2
FALSE
near(sqrt(2) ^ 2, 2)
TRUE
```

[b] Find all flights that: [2 points]

Had an arrival delay of three or more hours.

```
flights %>% filter(arr_delay>=3*60)
```

Flew to Dallas (DFW and DAL).

```
flights %>% filter(dest %in% c('DFW','DAL'))
```

Were operated by Southwest (WN) and American (AA).

```
flights %>% filter(carrier %in% c('WN','AA'))
```

Arrived more than two hours late but made up over 30 minutes in flight.

```
flights %>% filter(arr_delay > 2 * 60,(sched_arr_time - sched_dep_time) -
air_time  > 30)
```

Were delayed by at least an hour but made up over 30 minutes in flight.

```
flights %>% filter(dep_delay > 60,arr_delay > 60 ,(sched_arr_time -
sched_dep_time) - air_time  > 30)
```

[c] How could you use **`arrange( )`** to sort all missing values to the start? (Hint: use **`is.na( )`**). Show your sample code and use a short artificial tibble. [0.5 points]

```
flights %>% arrange(!is.na(flights))
```

[d] What is the statement **`select(tibble, var1, var2, everything())`** achieving? [0.25 points]

`everything()` selects all variable. The result of the above code would give you all variables in that Tibble instead of just var1 and var2

[e] What is the difference between **`mutate( )`** and **`transmute( )`**? [0.25 points]

`mutate()` adds new variables and preserves existing ones; **`transmute()`** adds new variables and drops existing ones. New variables overwrite existing variables of the same name. Variables can be removed by setting their value to NULL.

[f] Find the 10 most delayed flights using a ranking function. How do you handle ties? Read the documentation of **`min_rank( )`**. [0.5 points]

```
flights %>% filter(!is.na(arr_delay)) %>%
  mutate(ranking = rank(arr_delay, ties.method = "min")) %>%
  top_n(10)
```

Same with

```
flights %>% filter(!is.na(arr_delay)) %>%
  mutate(ranking = min_rank(arr_delay)) %>%
  top_n(10)
```

[g] Which carrier has the worst delays? Challenge: can you disentangle the effects of bad destination airports versus bad carriers? Why or why not? Hint: think about **`flights %>% group_by(carrier, dest) %>% summarize(n( ))`**. [0.75 point]

```
flights %>% group_by(carrier) %>%
  summarise(avg_dep_delay = mean(dep_delay[!is.na(dep_delay)]),
            avg_arr_delay = mean(arr_delay[!is.na(arr_delay)]))
   carrier avg_dep_delay avg_arr_delay
 * <chr>           <dbl>         <dbl>
 1 9E              16.7          7.38
 2 AA               8.59         0.364
 3 AS               5.80        -9.93
 4 B6              13.0          9.46
 5 DL               9.26         1.64
 6 EV              20.0         15.8
 7 F9              20.2         21.9
 8 FL              18.7         20.1
 9 HA               4.90        -6.92
10 MQ              10.6         10.8
11 OO              12.6         11.9
12 UA              12.1          3.56
13 US               3.78         2.13
14 VX              12.9          1.76
15 WN              17.7          9.65
16 YV              19.0         15.6
```

Carrier F9 has the worst delay on both arriving and departing.

[h] What time of day should you fly if you want to avoid delays as much as possible? [0.5 points]

```
flights %>% filter(dep_delay > 0) %>%
  group_by(hour) %>%
  summarise(avg_delay = mean(dep_delay),happend_times = n())
    hour avg_delay happend_times
```

```
 * <dbl>     <dbl>        <int>
 1     5      15.3          489
 2     6      24.2         5430
 3     7      24.1         4963
 4     8      29.9         6790
 5     9      29.7         5392
 6    10      32.6         4942
 7    11      32.5         5034
 8    12      32.3         6408
 9    13      33.5         8183
10    14      37.1         9257
11    15      38.8        11364
12    16      43.4        10699
13    17      45.3        12132
14    18      46.5        10636
15    19      51.1        10839
16    20      49.6         8633
17    21      50.3         5596
18    22      46.5         1184
19    23      38.0          461
```

From the given result above, I may want to fly in the early morning.

Task 2: Based on *Chapter 18 Pipes with magrittr* show the code how the piping operator `%>%` combines the three individual function calls **`functA( )`**, **`functB( )`** and **`functC( )`** internally into a joint function [2 points]

Sequence of function calls:

```
x <- functA(w)
y <- functB(x)
z <- functC(y)
```

```
z <- w %>% functA() %>% functB()%>%functC()
```

Task 3: Follow the modeling approach outlined in Boehmke and Greenwell's Chapter 2 *Modeling Process*. Your objective is to predict the probability of default in the dataset **`credit.csv`**. [6 points]

[a] Discuss your data splitting and resampling methods.[2 points]

```
# Stratified sampling with the rsample package
set.seed(123)
split <- initial_split(credit, prop = 0.7,
                       strata = "default")
default_train  <- training(split)
default_test   <- testing(split)
# Specify resampling strategy
cv <- trainControl(
  method = "repeatedcv",
  number = 10,
  repeats = 5
```

)
Use the stratified sampling for splitting our dataset to 70% for training and 30% for testing. As for resampling, we applied repeated 10 fold cross-validation methods for 5 times.

[b] Show the code and output of your analysis. [3 points]

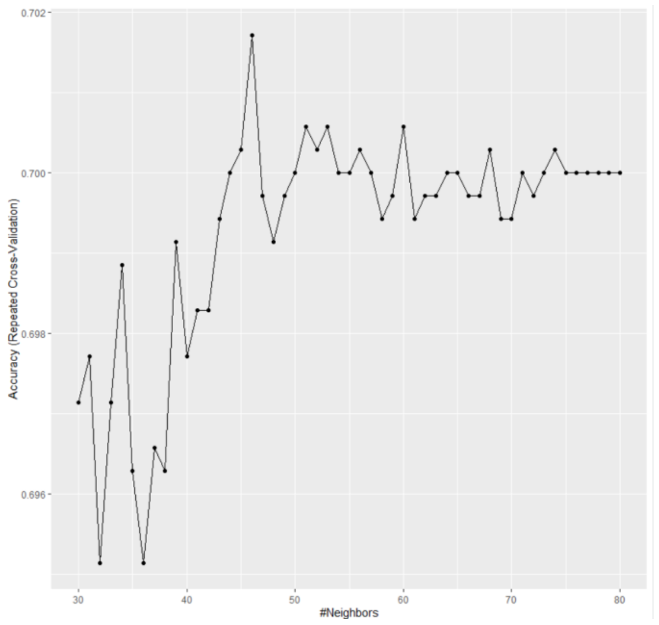
You can import the dataset with the statement

```
credit <-read.csv("Drive:\\Path\\credit.csv",
                  header = TRUE, stringsAsFactors = TRUE)
sapply(credit, is.factor)
```

Make sure to use proper a range to evaluate the hyperparameter k of your k nearest neighbor algorithm.

Because your prediction focuses on a classified data, the evaluation criterium needs to change from **metric = "RMSE"** to **metric = "Accuracy"**.

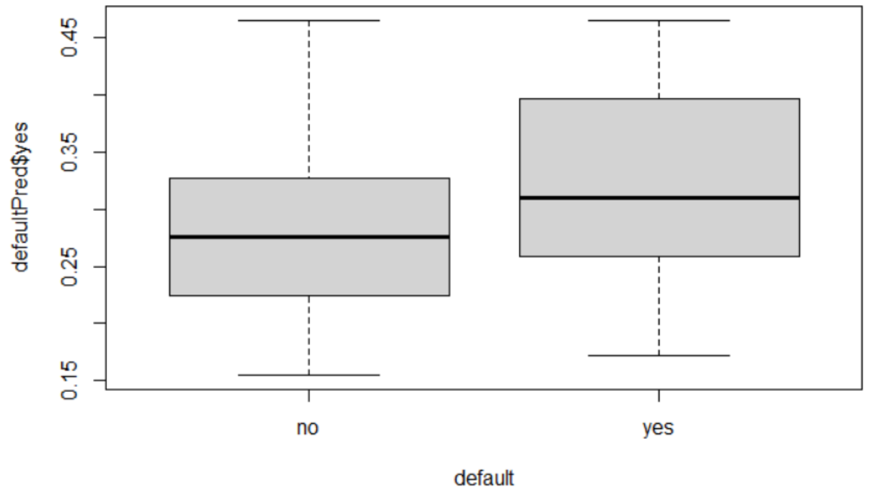
```
# Create grid of hyperparameter values
hyper_grid <- expand.grid(k = seq(30, 80, by = 1))

# Tune a knn model using grid search
knn_fit <- train(
  default ~ . ,
  data = default_train,
  method = "knn",
  trControl = cv,
  tuneGrid = hyper_grid,
  metric = "Accuracy"
)
knn_fit$bestTune
46
ggplot(knn_fit)
```



[c] Discuss the predictive quality of your model. [1 point]

You can evaluate the predicted probabilities with

```
defaultPred <- predict(knn_fit, default_test, type="prob")
plot(defaultPred$yes~default, data=default_test)
```



From the result, we would see the probability of more than 90% positive prediction from the model less than 0.5, which indicates our model does not give us an effective result. The alternative way to make it better is [1] do standardization and normalization for all metric variables, makes their contribution to our model are comparatively measurable, [2] Get rid of redundant information using the Principal component analysis.

Task 4: Follow the modeling approach outlined in Boehmke and Greenwell's Chapter 3 *Feature and Target Engineering*. Use the dataset **tractShp** in the package **TexMix**. [3 points]

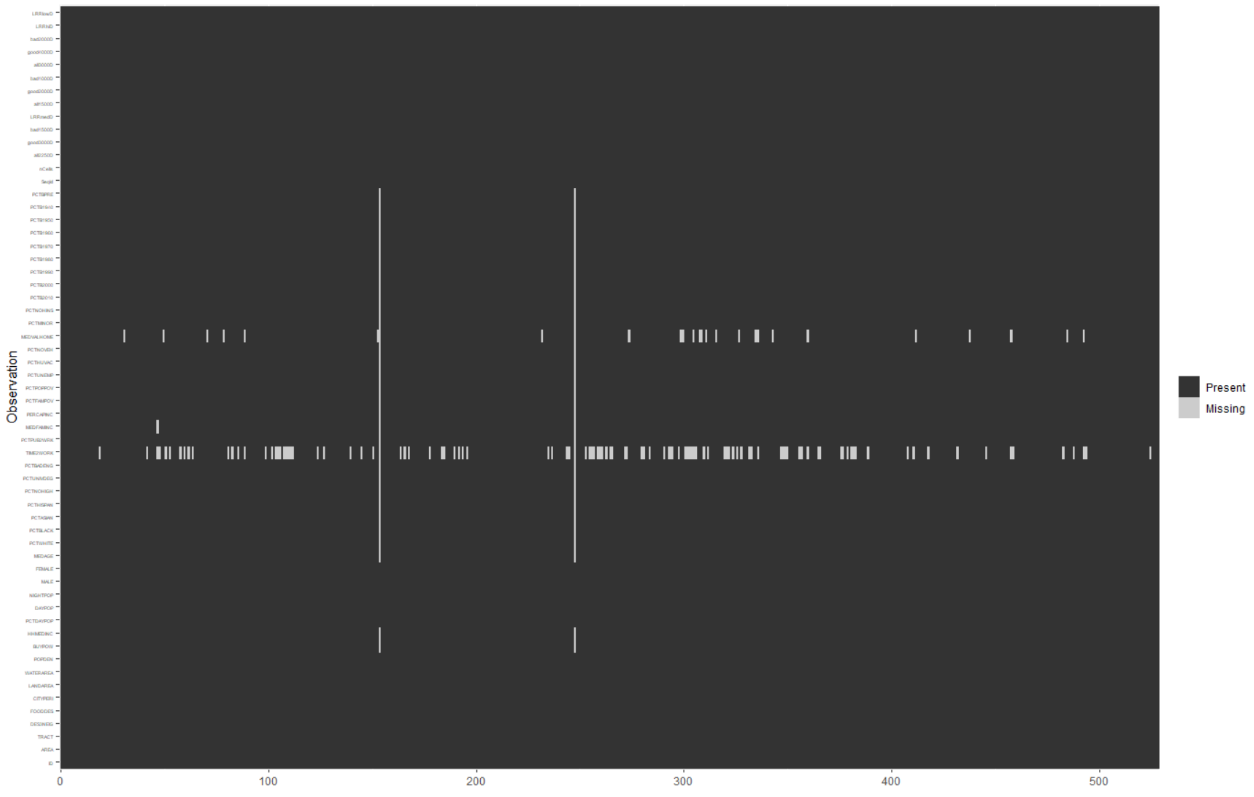
You can extract the attribute data-frame from the shape file with the statement **tractDf <- as.data.frame(tractShp)**.

Show all code and output.

[a] Evaluate the missingness of all variables in the dataset and impute their missing values. Should there be any census tracts that need to be excluded from the analysis on logical grounds? [2 points]

```
tractDf %>% is.na() %>%
  apply(1,sum) %>%
  table()
  0   1   2  31
400 120   7   2
tractDf %>% is.na() %>%
  apply(2,sum) %>%
  table()
  0   2   3  27 110
 28  28   1   1   1
```

```
which(apply(is.na(tractDf), 2, sum)==110)
TIME2WORK
       25

tractDf %>% is.na() %>%
  melt() %>%
  ggplot(aes(x = Var2,y = Var1,fill=value)) +
  geom_raster() +
  coord_flip()+
  scale_y_continuous(NULL, expand = c(0,0)) +
  scale_fill_grey(name = '',labels = c("Present",'Missing'))+
  xlab('Observation') +
  theme(axis.text.y = element_text(size = 4))
```



According to our analysis result, there are two census tracts have 31 missing value among total 57 recorded features. We should eliminate them from our dataset since we do not have enough information to make an inference from them. Besides, the **time2work** variable has 110 missing values over 529 observations, we should also delete it from our samples.

[b] Standardize all metric variables. [1 point]

```
library(recipes)
rec <- recipe(PCTB2010 ~ .,data = tractDf)
rec %>%
  step_center(all_numeric()) %>%
  step_scale(all_numeric())
```