Mitigating Churn

Victoria Espinola

2022-10-17

A1. According to the data dictionary churn rate can be has high as 25% per year and cost 10 times more to aquire new customers than it does to retain an existing customer. Stakeholders often try to mitigate the cost by early intervention inncentive techniques aiming to retain existing customers. The k-nearest neighbor or KNN process uses Euclidean distance between an observation's k nearest neighbors to classify an observation as either a customer that will churn or be retained.

A2. The goal of the analysis is to determine if we can get beat a baseline model accuracy of 73% ,obtained from predicting all customers were retained since 6564 customers were retained out of the 8950 observations remaining after cleaning. If this is the case the model can then be used for early intervention techniques such as financial incentives for customers that are predicted to churn or other intervention techniques chosen by the business stakeholders.

B1. K nearest neighbor or KNN is an algorithm that classifies an observation based on information from "nearby" observations. The algorithm calculates the nearness using the euclidean distance between observations. Once the nearest neighbors are found the majority class from the k nearest observations is used as the predicted class for the new observation.The outcome of this method will be used to determine which customers may churn.

B2. One assumption of the KNN classification method is that all variables are numeric so that they can be used to calculate distance.

B3. The following packages were used for this project and their justifications are given in the comments below.

```
library(tidyr)  #This package is for tidying the data and making outputs more readable.
library(tidymodels) #This package assists with train/test split.
library(class) #This package supports the KNN algorithm.
library(janitor) #This package helps to prepare the data
library(caret) #This package helps to prepare the data by dummying each categorical data point
library(pROC) #The package helps to create the ROC curve and calculate the AUC value
source("cleaner_copy.R") #This is a cleaning script I wrote to clean the given data set.
```

C1. One data processing goal is to eliminate outliers from the data set. Since KNN relies on distance, all outliers left in the data set heavily influence the predictions.

C2. The following shows the initial data set being brought in and eliminates the uid, case_order, lat, lng, state, county, and time zone variables. The uid and case_order variables were selected for elimination because they are used primarily for identifying customers and not customer attributes or habits. Additionally, lat, lng, state, county, and time zone seemed redundant, so zip was kept in the final data set.

```
churn_data <- read.csv("churn_clean.csv", header = TRUE) %>%
  janitor::clean_names() %>%
  churn_cleaning() %>%
  select(-uid, -case_order, -lat, -lng, -state, -county,-time_zone)
```

The following is the summary statistic of each variable and identifies if the variable is continuous or categorical.

```
skimr::skim(churn_data)
```

Data summary

Name	churn_data
Number of rows	8950
Number of columns	38
________________________	
Column type frequency:	
factor	19
numeric	19
________________________	
Group variables	None

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
area	0	1	FALSE	3	Sub: 2994, Rur: 2984, Urb: 2972

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
marital	0	1	FALSE	5	Div: 1866, Wid: 1828, Sep: 1804, Nev: 1753
gender	0	1	FALSE	3	Fem: 4493, Mal: 4250, Non: 207
churn	0	1	FALSE	2	No: 6564, Yes: 2386
techie	0	1	FALSE	2	No: 7452, Yes: 1498
contract	0	1	FALSE	3	Mon: 4868, Two: 2193, One: 1889
port_modem	0	1	FALSE	2	No: 4616, Yes: 4334
tablet	0	1	FALSE	2	No: 6280, Yes: 2670
internet_service	0	1	FALSE	3	Fib: 3939, DSL: 3092, Non: 1919
phone	0	1	FALSE	2	Yes: 8111, No: 839
multiple	0	1	FALSE	2	No: 4792, Yes: 4158
online_security	0	1	FALSE	2	No: 5750, Yes: 3200
online_backup	0	1	FALSE	2	No: 4913, Yes: 4037
device_protection	0	1	FALSE	2	No: 5034, Yes: 3916
tech_support	0	1	FALSE	2	No: 5603, Yes: 3347
streaming_tv	0	1	FALSE	2	No: 4536, Yes: 4414
streaming_movies	0	1	FALSE	2	No: 4557, Yes: 4393
paperless_billing	0	1	FALSE	2	Yes: 5259, No: 3691
payment_method	0	1	FALSE	4	Ele: 3045, Mai: 2049, Ban: 1999, Cre: 1857

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
zip	0	1	48510.38	26857.24	1005.00	26288.00	48388.00	70361.75	99401.00	▇▇▇▇▆
children	0	1	1.94	1.89	0.00	0.00	1.00	3.00	8.00	▇▅▂▁▁
age	0	1	53.16	20.63	18.00	35.00	53.00	71.00	89.00	▇▇▇▇▇
income	0	1	38329.40	25123.53	348.67	19041.12	32778.48	52280.44	124025.10	▇▇▃▂▁
outage_sec_perweek	0	1	10.01	2.93	1.14	8.03	10.02	11.96	18.85	▁▅▇▅▁
email	0	1	12.02	3.01	3.00	10.00	12.00	14.00	21.00	▁▆▇▆▁
contacts	0	1	0.94	0.90	0.00	0.00	1.00	2.00	3.00	▇▇▁▅▂
yearly_equip_failure	0	1	0.37	0.58	0.00	0.00	0.00	1.00	2.00	▇▁▃▁▁
tenure	0	1	34.42	26.45	1.01	7.89	29.77	61.39	72.00	▇▂▁▂▆
monthly_charge	0	1	172.78	42.99	79.98	139.98	167.48	202.44	290.16	▂▇▆▃▁
bandwidth_gb_year	0	1	3379.46	2185.20	155.51	1228.08	3120.63	5579.37	7158.98	▇▃▁▆▅
item1	0	1	3.47	1.01	1.00	3.00	3.00	4.00	6.00	▂▇▇▃▁
item2	0	1	3.49	1.02	1.00	3.00	3.00	4.00	6.00	▂▇▇▃▁
item3	0	1	3.47	1.02	1.00	3.00	3.00	4.00	6.00	▂▇▇▃▁
item4	0	1	3.49	1.02	1.00	3.00	3.00	4.00	6.00	▂▇▇▃▁
item5	0	1	3.49	1.02	1.00	3.00	3.00	4.00	6.00	▂▇▇▃▁

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
item6	0	1	3.48	1.02	1.00	3.00	3.00	4.00	6.00	▃▇▇▃▁
item7	0	1	3.50	1.02	1.00	3.00	3.00	4.00	6.00	▃▇▇▃▁
item8	0	1	3.48	1.02	1.00	3.00	3.00	4.00	6.00	▃▇▇▃▁

C3. To prepare the data, first one-hot encoding was used for factor variables, the dummied variables were put into a new data frame and binded back column wise with numeric variables, and finally the non-dummied categorical variables were removed.

```
#Dummy variables using one-hot encoding.
one_hot_dummy <- dummyVars(
  " ~.",
  data = (churn_data %>%
            select(where(is.factor) & !contains("churn")) #churn is not included in one-hot encoding since
all non-dummied variables will be removed.
          )
  )
dummy_df <- data.frame(
  predict(
    one_hot_dummy,
    newdata = (churn_data %>%
                 select(where(is.factor) & !contains("churn")) #creates new df with dummied variables.
               )
    )
)

final_data_set <- dummy_df %>% bind_cols(churn_data) %>% #put together numeric with dummied variables
  select(!where(is.factor), churn)%>%  #remove all non-dummied variables
  janitor::clean_names()


#normalizing numeric data, makes scale between 0 and 1

scaled_ds <- final_data_set %>%
    mutate(
    across(where(is.numeric), ~(.x-min(.x))/(max(.x)-min(.x))
      )
    )
```

C4. The final data set has been normalized and dummy variables have categorical features have been one hot encoded.

```
skimr::skim(scaled_ds)
```

Data summary

Name	scaled_ds
Number of rows	8950
Number of columns	65
________________________	
Column type frequency:	
factor	1
numeric	64
________________________	
Group variables	None

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
churn	0	1	FALSE	2	No: 6564, Yes: 2386

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
area_rural	0	1	0.33	0.47	0	0.00	0.00	1.00	1	▇▁▁▁▃
area_suburban	0	1	0.33	0.47	0	0.00	0.00	1.00	1	▇▁▁▁▃
area_urban	0	1	0.33	0.47	0	0.00	0.00	1.00	1	▇▁▁▁▂
marital_divorced	0	1	0.21	0.41	0	0.00	0.00	0.00	1	▇▁▁▁▂
marital_married	0	1	0.19	0.39	0	0.00	0.00	0.00	1	▇▁▁▁▂
marital_never_married	0	1	0.20	0.40	0	0.00	0.00	0.00	1	▇▁▁▁▂
marital_separated	0	1	0.20	0.40	0	0.00	0.00	0.00	1	▇▁▁▁▂
marital_widowed	0	1	0.20	0.40	0	0.00	0.00	0.00	1	▇▁▁▁▂
gender_female	0	1	0.50	0.50	0	0.00	1.00	1.00	1	▇▁▁▁▇
gender_male	0	1	0.47	0.50	0	0.00	0.00	1.00	1	▇▁▁▁▇
gender_nonbinary	0	1	0.02	0.15	0	0.00	0.00	0.00	1	▇▁▁▁▁
techie_no	0	1	0.83	0.37	0	1.00	1.00	1.00	1	▂▁▁▁▇
techie_yes	0	1	0.17	0.37	0	0.00	0.00	0.00	1	▇▁▁▁▂
contract_month_to_month	0	1	0.54	0.50	0	0.00	1.00	1.00	1	▇▁▁▁▇
contract_one_year	0	1	0.21	0.41	0	0.00	0.00	0.00	1	▇▁▁▁▂
contract_two_year	0	1	0.25	0.43	0	0.00	0.00	0.00	1	▇▁▁▁▂
port_modem_no	0	1	0.52	0.50	0	0.00	1.00	1.00	1	▇▁▁▁▇
port_modem_yes	0	1	0.48	0.50	0	0.00	0.00	1.00	1	▇▁▁▁▇
tablet_no	0	1	0.70	0.46	0	0.00	1.00	1.00	1	▃▁▁▁▇
tablet_yes	0	1	0.30	0.46	0	0.00	0.00	1.00	1	▇▁▁▁▃
internet_service_dsl	0	1	0.35	0.48	0	0.00	0.00	1.00	1	▇▁▁▁▅
internet_service_fiber_optic	0	1	0.44	0.50	0	0.00	0.00	1.00	1	▇▁▁▁▆
internet_service_none	0	1	0.21	0.41	0	0.00	0.00	0.00	1	▇▁▁▁▂
phone_no	0	1	0.09	0.29	0	0.00	0.00	0.00	1	▇▁▁▁▁
phone_yes	0	1	0.91	0.29	0	1.00	1.00	1.00	1	▁▁▁▁▇
multiple_no	0	1	0.54	0.50	0	0.00	1.00	1.00	1	▇▁▁▁▇
multiple_yes	0	1	0.46	0.50	0	0.00	0.00	1.00	1	▇▁▁▁▇
online_security_no	0	1	0.64	0.48	0	0.00	1.00	1.00	1	▅▁▁▁▇
online_security_yes	0	1	0.36	0.48	0	0.00	0.00	1.00	1	▇▁▁▁▅
online_backup_no	0	1	0.55	0.50	0	0.00	1.00	1.00	1	▆▁▁▁▇
online_backup_yes	0	1	0.45	0.50	0	0.00	0.00	1.00	1	▇▁▁▁▆
device_protection_no	0	1	0.56	0.50	0	0.00	1.00	1.00	1	▆▁▁▁▇
device_protection_yes	0	1	0.44	0.50	0	0.00	0.00	1.00	1	▇▁▁▁▆

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
tech_support_no	0	1	0.63	0.48	0	0.00	1.00	1.00	1	▅▁▁▁▇
tech_support_yes	0	1	0.37	0.48	0	0.00	0.00	1.00	1	▇▁▁▁▅
streaming_tv_no	0	1	0.51	0.50	0	0.00	1.00	1.00	1	▇▁▁▁▇
streaming_tv_yes	0	1	0.49	0.50	0	0.00	0.00	1.00	1	▇▁▁▁▇
streaming_movies_no	0	1	0.51	0.50	0	0.00	1.00	1.00	1	▇▁▁▁▇
streaming_movies_yes	0	1	0.49	0.50	0	0.00	0.00	1.00	1	▇▁▁▁▇
paperless_billing_no	0	1	0.41	0.49	0	0.00	0.00	1.00	1	▇▁▁▁▆
paperless_billing_yes	0	1	0.59	0.49	0	0.00	1.00	1.00	1	▆▁▁▁▇
payment_method_bank_transfer_automatic	0	1	0.22	0.42	0	0.00	0.00	0.00	1	▇▁▁▁▂
payment_method_credit_card_automatic	0	1	0.21	0.41	0	0.00	0.00	0.00	1	▇▁▁▁▂
payment_method_electronic_check	0	1	0.34	0.47	0	0.00	0.00	1.00	1	▇▁▁▁▅
payment_method_mailed_check	0	1	0.23	0.42	0	0.00	0.00	0.00	1	▇▁▁▁▂
zip	0	1	0.48	0.27	0	0.26	0.48	0.70	1	▇▇▇▇▆
children	0	1	0.24	0.24	0	0.00	0.12	0.38	1	▇▅▂▁▁
age	0	1	0.50	0.29	0	0.24	0.49	0.75	1	▇▇▇▇▇
income	0	1	0.31	0.20	0	0.15	0.26	0.42	1	▇▇▂▂▁
outage_sec_perweek	0	1	0.50	0.17	0	0.39	0.50	0.61	1	▁▅▇▅▁
email	0	1	0.50	0.17	0	0.39	0.50	0.61	1	▁▆▇▆▁
contacts	0	1	0.31	0.30	0	0.00	0.33	0.67	1	▇▇▁▅▂
yearly_equip_failure	0	1	0.19	0.29	0	0.00	0.00	0.50	1	▇▁▂▁▁
tenure	0	1	0.47	0.37	0	0.10	0.41	0.85	1	▇▂▁▃▆
monthly_charge	0	1	0.44	0.20	0	0.29	0.42	0.58	1	▂▇▆▃▁
bandwidth_gb_year	0	1	0.46	0.31	0	0.15	0.42	0.77	1	▇▃▁▆▅
item1	0	1	0.49	0.20	0	0.40	0.40	0.60	1	▃▇▇▃▁
item2	0	1	0.50	0.20	0	0.40	0.40	0.60	1	▃▇▇▃▁
item3	0	1	0.49	0.20	0	0.40	0.40	0.60	1	▃▇▇▃▁
item4	0	1	0.50	0.20	0	0.40	0.40	0.60	1	▃▇▇▃▁
item5	0	1	0.50	0.20	0	0.40	0.40	0.60	1	▃▇▇▃▁
item6	0	1	0.50	0.20	0	0.40	0.40	0.60	1	▃▇▇▃▁
item7	0	1	0.50	0.20	0	0.40	0.40	0.60	1	▃▇▇▃▁
item8	0	1	0.50	0.20	0	0.40	0.40	0.60	1	▃▇▇▃▁

```
#write.csv(scaled_ds, "scaled_ds.csv")
```

D1. The following is setting a seed to have reproducible results and will split the data into testing and training set. (DataCamp and Delpiaz)

```
set.seed(123) #Seed is to maintain consistant results and will help keep tie breakers the same as tie breaker
s are random and not based on algorithm.
```

```
splits <- initial_split(scaled_ds)
train_data <- training(splits) %>% select(-churn)
churn_train <- training(splits)$churn
test_data <- testing(splits) %>% select(-churn)
churn_test <- testing(splits)$churn
```

```
#code that writes the csv for final submission.
#write.csv(training(splits), "train_data.csv")
#write.csv(testing(splits), "test_data.csv")
```

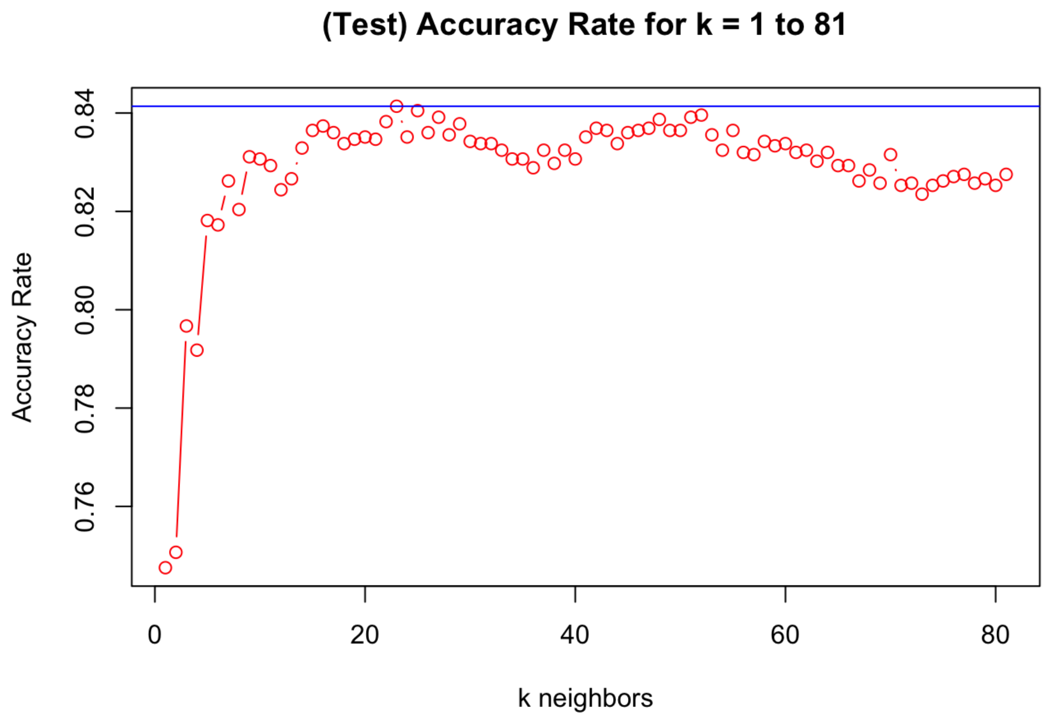
D2. The model will be assessed by its accuracy but first an appropriate value of k should be found. The following loop will take values between 1 and the square root of the number of values in the training set (DataCamp course)

D3. The follow code shows the KNN technique starting with selecting the best value of k neighbors to consider based on accuracy.

```
#Setting K value range
k = 1:sqrt(nrow(train_data))

#Preparing to find accuracy for the loop
accu_k = rep(x = 0, times= length(k))

#Accuracy function will be calculated from when the actual and predicted values are the same.
calc_class_acc  = function(actual, prediction) {
  mean(actual == prediction)
}

#Loop for determining the best k value, at which value is the accuracy the largest.
for (t in seq_along(k)) {
  pred = knn(train = train_data,
             test = test_data,
             cl = churn_train,
             k = k[t])
  accu_k[t] = calc_class_acc(churn_test, pred)
}
```

The following shows the accuracy of the model based on various k values.

```
plot(accu_k, type = "b", col = "red",
     xlab = "k neighbors", ylab = "Accuracy Rate",
     main = "(Test) Accuracy Rate for k = 1 to 81")

abline(h = max(accu_k), col = "blue")
```



The graph shows that the model reaches its best accuracy between 20 and 40, this value is optimial since it will produce a model that will not be overfit. The following are the calculations used to determine the exact point location.

```
best_k = min(which(accu_k == max(accu_k))) #finding the first instance with the highest accuracy rate and low
est value of k.
best_k #prints values of k.
```

```
## [1] 23
```

The final KNN model is shown below using the k value of 23.

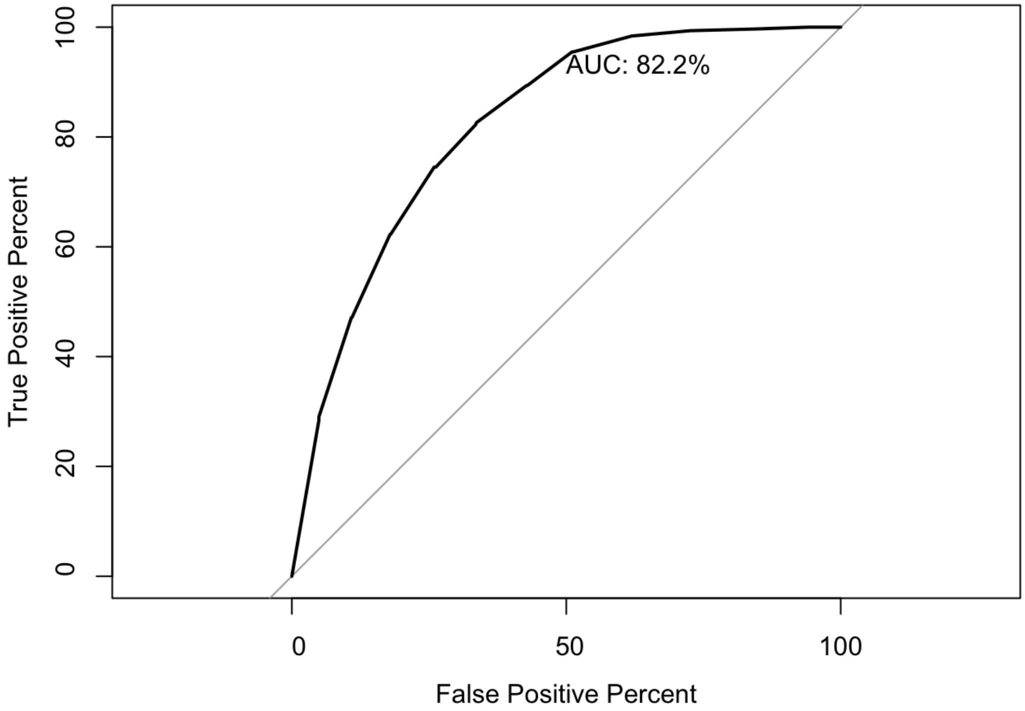
```
final_knn_prob <- knn(train = train_data,
                 test = test_data,
                 k = 23,
                 cl = churn_train,
                 prob = TRUE)
```

E1. The final classification model that has a k-value of 23 has an accuracy of 84% and AUC value is 82.2%. That means that the model misclassifies 16% of the data that it is given.

```
conf_matrix <- table(final_knn_prob, churn_test)
conf_matrix
```

```
##                churn_test
## final_knn_prob   No  Yes
##             No 1613  315
##             Yes   42  268
```

```
roc(final_knn_prob, attributes(final_knn_prob)$prob, plot = TRUE, legacy.axes = TRUE, percent = TRUE,
    xlab = "False Positive Percent", ylab = "True Positive Percent", print.auc = TRUE,print.auc.y = 95)
```



```
##
## Call:
## roc.default(response = final_knn_prob, predictor = attributes(final_knn_prob)$prob,     percent = TRUE, pl
ot = TRUE, legacy.axes = TRUE, xlab = "False Positive Percent",     ylab = "True Positive Percent", print.auc
= TRUE, print.auc.y = 95)
##
## Data: attributes(final_knn_prob)$prob in 1928 controls (final_knn_prob No) > 310 cases (final_knn_prob Ye
s).
## Area under the curve: 82.24%
```

E2. The model incorrectly classifiies 16% of the data. Specifically, 357 customers were incorrectly classified, with the largest group being 315 customers that were predicted not to churn, but actually did churn. Similarly, the other 42 customers that were misclassified were predicted to churn, but they did not. The KNN algorithm used from the class package did not allow for the distance metric used so a different metric may lead to better results.

E3. One limitation of my data analysis method is that the data is skewed and has more observations of customers who do not churn compared to customers who do churn. A follow up model should include a resampling technique that will help even out the skewed data and may lead to better predictions.

E4. The purpose of the analysis was to help stakeholders decide whether to move forward with implementing a model that would improve churn prediction accuracy. Since our model accuracy is 84%, better than the 73% accuracy of the baseline model that comes from predicting all customers wil stay, the model could prove useful in helping the business in their identification of churning customers and implementation of an intervention program.

H. Acknowledgement and Citations

1. Brett Lantz. https://www.datacamp.com/users/sign_in?redirect=http%3A%2F%2Fapp.datacamp.com%2Flearn%2Fcourses%2Fsupervised-learning-in-r-classification (https://www.datacamp.com/users/sign_in?redirect=http%3A%2F%2Fapp.datacamp.com%2Flearn%2Fcourses%2Fsupervised-learning-in-r-classification)

2. Dalpiaz, D. (2020, October 28). Chapter 12 k-Nearest Neighbors | R for Statistical Learning. Retrieved October 14, 2022, from https://daviddalpiaz.github.io/r4sl/knn-class.html (https://daviddalpiaz.github.io/r4sl/knn-class.html)