Improving Churn Model with Random Forest

Victoria Espinola

2022-10-21

A1. According to the churn data dictionary, the churn rate for telecommunications customers can be as high as 25% per year and it can cost up to 10 times as much to acquire a new customer than retaining an existing customer. A KNN model was applied to the customer churn data set in task one and had an accuracy rate of 84% for the outcome churn. While this anaylsis was enough to support moving forward with early intervention to retain at-risk customers, the goal of this analysis is to improve upon that classification analysis by appling a random forest model.

A2. The goal of this analysis is to improve upon the KNN model and create a random forest model that will have a better accuracy than the KNN model. If the random forest model has an accuracy higher than 84%, then it will be an improvement on the KNN model. The final model will be given to stakeholders to isolate target customers for intervention and show the predictor variables that have the most effect on churn outcome.

B1. A random forest method trains a model based on creating individual decision trees from randomly selected subsets of the predictor variables from the training data. The collection of trees is then put together as an ensemble and the majority class is used to generate the predictive class. The ensemble can be improved upon by tuning the hyperparameters, and thus can increase the classification accuracy (Applied Predictive Modeling, M. Kuhn, section 8.5). According to section 4 of Machine Learning with Tree-Based Models in R, random forest is a good “out-of-the-box” performer, hence, the final model is expected to perform better than the KNN model from task 1 (DataCamp).

B2. One assumption of the random forest method is that low variance variables need to be removed to avoid overfitting the model (Datacamp, Machine Learning with Caret, M. Kuhn). One way to recongnize overfitting is observing that the model has a low in sample error but high out of sample error.

B3. The libaries used for this analysis include, tidyverse, tidymodels, janitor, and vip. These libraries were used because they contain the function for reading in the data, cleaning and preparing the data, creating a recipe, analyzing the data, and assessing the models performance. The list of libraries are given in the code below with comments on how they contributed to the anaylsis.

```
library(tidyverse) #tidyverse meta package containing useful tools for preparation, dply
r, ggplot2, and readr
library(tidymodels) #tidymodels meta package containing useful tools for the analysis, s
uch as tunes, yardstick, recipes, workflow and dials.
library(janitor) #janitor cleans names
library(vip) #vip is used to find the most important variables.
```

C1. In order to use the random forest method for classification, the data must have all low variance variables removed, helping the model not overfit the data. While tree models are robust to outliers, when left in the data set, they can increase the likilhood of overfitting the data as the model tend to learn the noise within the data set. Therefore, the three main goals of the preparation are 1) removing all low variance variables, 2) normalize the observations using step normalize, and 3) impute all missing values with the mean.

C2. Since one of the goals of the data preparation steps include removing all zero variance variables, the entire given data set will be used to classify churn outcomes. All calculations needed for data preparation were completed with the recipe function from tidymodels and no manual calculations were computed. Additionally, the comments show the purpose for each step of the recipe.

C3. The code below shows the steps for data preparation, the comments annotate what is being completed in each step. The process for data preparation can be summed up as, bringing the data set and changing character variables to factor variables, creating a seed for reproducibility, splitting the data into training and testing sets, and finally creating the recipe for feature engineering and data prep as seen in the comments. The code concludes with a summary of all the variables and their distrubutions after recipe steps were applied.

```
churn_data <- read.csv("churn_clean.csv", header = TRUE) %>%
  mutate_if(is.character,as.factor) %>%                    #bringing in the data, chang
ing nomial variables as factors
  janitor::clean_names()                                   #standardizing naming conven
tions

# Create training and testing splits ----
set.seed(123)
splits <- initial_split(churn_data, prop = 0.80, strata = churn)  #initial data split, s
trata ensures each set will have the
same proportion of churn observations.

train_churn <- training(splits)
test_churn  <- testing(splits)


# Create feature engineering recipe for initial model ----
recipe_spec <- recipe(churn ~., data = train_churn ) %>%
  #impute the missing variables with the mean
  step_impute_mean(all_numeric_predictors()) %>%
  #removed any identification variables that do not add insight into the classification
 model.
  step_rm(county, state, time_zone, lat, lng, zip) %>%
  #removed all no variance variable from all predictors
  step_nzv(all_predictors())%>%
  #group together all nominal predictors into an other category when they are retained f
rom low variance,
  #but account for 0.5% of the data
  step_other(all_nominal_predictors(),threshold = 0.005) %>%
  #normalize the numeric data; this step is important to reduce any outliers in the data
set.
  step_normalize(all_numeric_predictors())

prepped_data <- recipe_spec %>% prep() %>% bake(new_data = churn_data) %>% janitor::clea
n_names()
skimr::skim(prepped_data)
```

Data summary

Name	prepped_data
Number of rows	10000
Number of columns	44
_____	
Column type frequency:	
factor	24
numeric	20

Group variables	None

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
customer_id	0	1	FALSE	2	oth: 9999, A04: 1
interaction	0	1	FALSE	2	oth: 9999, 000: 1
uid	0	1	FALSE	2	oth: 9999, 000: 1
city	0	1	FALSE	2	oth: 9966, Hou: 34
area	0	1	FALSE	3	Sub: 3346, Rur: 3327, Urb: 3327
job	0	1	FALSE	2	oth: 9974, Sal: 26
marital	0	1	FALSE	5	Div: 2092, Wid: 2027, Sep: 2014, Nev: 1956
gender	0	1	FALSE	3	Fem: 5025, Mal: 4744, Non: 231
techie	0	1	FALSE	2	No: 8321, Yes: 1679
contract	0	1	FALSE	3	Mon: 5456, Two: 2442, One: 2102
port_modem	0	1	FALSE	2	No: 5166, Yes: 4834
tablet	0	1	FALSE	2	No: 7009, Yes: 2991
internet_service	0	1	FALSE	3	Fib: 4408, DSL: 3463, Non: 2129
phone	0	1	FALSE	2	Yes: 9067, No: 933
multiple	0	1	FALSE	2	No: 5392, Yes: 4608
online_security	0	1	FALSE	2	No: 6424, Yes: 3576
online_backup	0	1	FALSE	2	No: 5494, Yes: 4506
device_protection	0	1	FALSE	2	No: 5614, Yes: 4386
tech_support	0	1	FALSE	2	No: 6250, Yes: 3750
streaming_tv	0	1	FALSE	2	No: 5071, Yes: 4929
streaming_movies	0	1	FALSE	2	No: 5110, Yes: 4890
paperless_billing	0	1	FALSE	2	Yes: 5882, No: 4118
payment_method	0	1	FALSE	4	Ele: 3398, Mai: 2290, Ban: 2229, Cre: 2083
churn	0	1	FALSE	2	No: 7350, Yes: 2650

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
case_order	0	1	0.00	1.00	-1.73	-0.86	0.00	0.87	1.74	
population	0	1	0.00	1.00	-0.68	-0.63	-0.48	0.23	7.05	
children	0	1	0.00	1.01	-0.98	-0.98	-0.51	0.43	3.71	
age	0	1	0.01	1.00	-1.69	-0.87	0.00	0.87	1.75	
income	0	1	0.00	1.00	-1.40	-0.73	-0.23	0.48	7.76	
outage_sec_perweek	0	1	0.00	1.00	-3.31	-0.67	0.00	0.66	3.75	
email	0	1	0.00	1.00	-3.65	-0.67	-0.01	0.65	3.63	
contacts	0	1	0.00	1.00	-1.01	-1.01	0.01	1.02	6.10	
yearly_equip_failure	0	1	0.00	1.00	-0.63	-0.63	-0.63	0.94	8.79	
tenure	0	1	0.00	1.00	-1.27	-1.00	0.04	1.02	1.42	
monthly_charge	0	1	0.00	1.00	-2.16	-0.76	-0.12	0.66	2.74	
bandwidth_gb_year	0	1	0.00	1.00	-1.48	-0.98	-0.05	1.01	1.73	
item1	0	1	-0.01	1.00	-2.41	-0.48	-0.48	0.48	3.37	
item2	0	1	-0.01	1.00	-2.43	-0.49	0.47	0.47	3.37	
item3	0	1	-0.01	1.00	-2.42	-0.48	-0.48	0.49	4.36	
item4	0	1	0.00	1.00	-2.42	-0.48	-0.48	0.49	3.41	
item5	0	1	0.00	1.00	-2.43	-0.48	-0.48	0.49	3.42	
item6	0	1	0.00	1.00	-2.42	-0.48	-0.48	0.48	4.35	
item7	0	1	0.01	1.00	-2.43	-0.49	0.48	0.48	3.40	
item8	0	1	0.00	0.99	-2.41	-0.48	-0.48	0.49	4.36	

C4. The code below shows the script for creating the csv file with all the data to be uploaded for the final submission.

```
write.csv(prepped_data, "prepped_data.csv" )
```

D1. In the previous chunk of code, the character variables have been converted to factors and the data had an initial 80/20 split with a stratification on the churn outcome. From the initial split the training and testing sets were created. The code below applies the preparation reciepe steps to the split data and outputs the new preppred_train and prepped_test data sets that will be used for analysis.

```
# Prep data, clean the names
prepped_train_data <- recipe_spec %>% prep() %>% bake(new_data = train_churn) %>% janito
r::clean_names()
prepped_test_data <- recipe_spec %>% prep() %>% bake(new_data = test_churn) %>% janito
r::clean_names()
```

Now the prepped data will be writen as CSV files for attachment to the final submission.

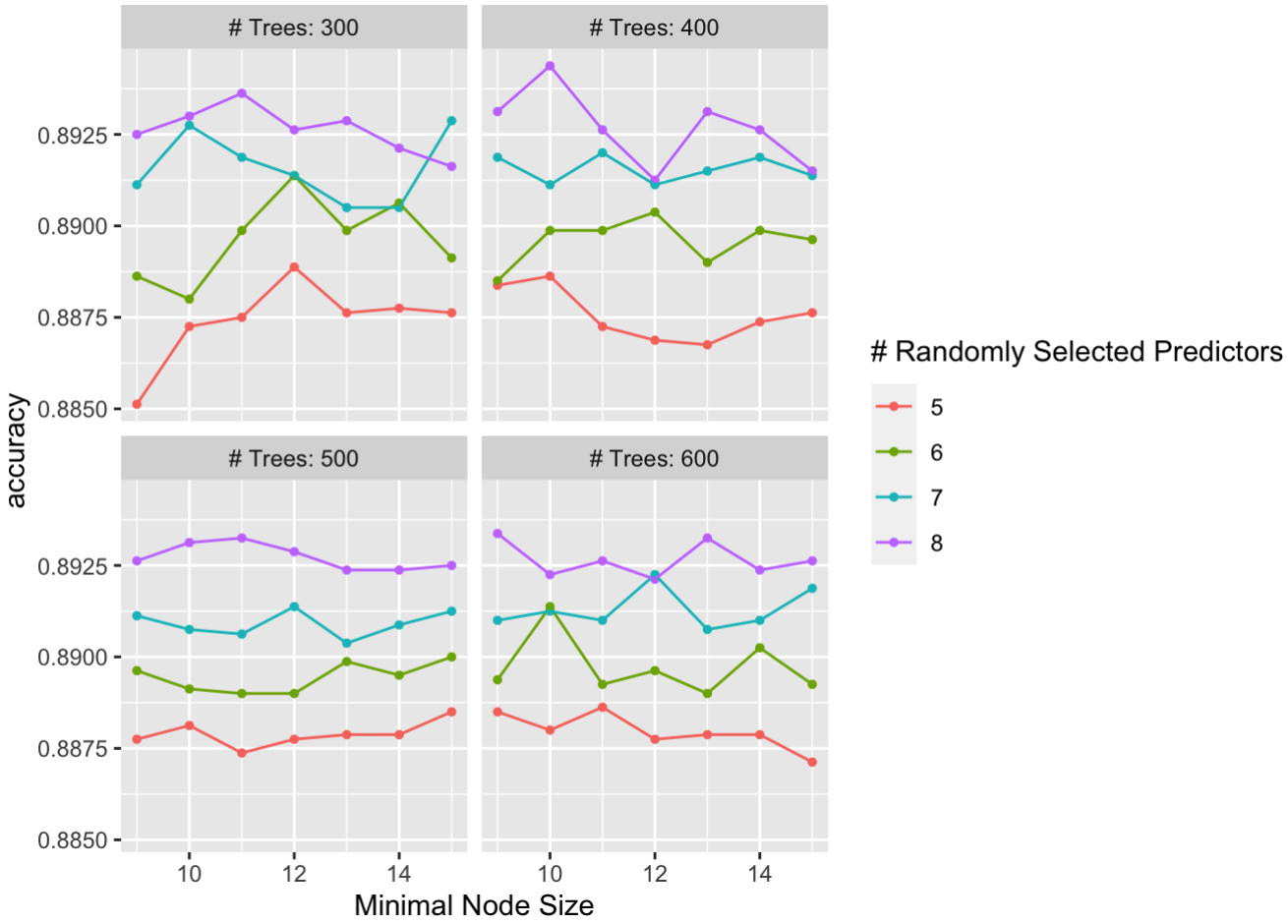
```
#Save prepped data as CSV
write.csv(prepped_train_data, "prepped_train_data.csv" )
write.csv(prepped_test_data, "prepped_test_data.csv")
```

D2. The following analysis technique was implemented from the DataCamp course, "Machine Learning with Tree Models, section 4"; start with the spec model with tuning parameters, set up the number of folds, and create a tuning grid, then use the grid results to find the best parameters, in this instance accuracy will be used to assess the best tuned model parameters. Finally, the tuning specs are passed to the model spec to finalize the parameters the model will use when trained. The model accuracy with the finalized hyperparameters will be assessed using 8-fold cross validation on the training set and then if satisfactory the model will be fit on the entire training set.

***The following code chunk will not be run during demonstration due to processing time.**

```
#Set tuning parameters for random forest classification.
rand_forest_spec <- rand_forest(
  mtry  = tune(),
  trees = tune(),
  min_n = tune()
) %>%
  set_mode("classification") %>%
  set_engine("ranger", importance = "permutation") #needed to create variable importance
plot

#set seed for reproducibility for folds and tuning grid because both having a random ele
ment.
set.seed(123)

#set up tuning grid
test_grid <- tidyr::crossing(
  mtry  = 5:8, #best practice is sqrt of number of variables, sqrt(50)~ 7
  trees = seq(300,600,100), #best practice is 500 trees from ranger parsnip doc.
  min_n = 9:15 # best practice is value of 10 from ranger parsnip doc.
)

set.seed(123)
doParallel::registerDoParallel() #improve the time for the model to run.
tune_results <- tune_grid(rand_forest_spec,
                          recipe_spec,
                          resamples = tree_folds,
                          grid      = test_grid,
                          metrics   = metric_set(accuracy) #parameters optimized by accu
racy.
                          )

#Select best tune results from the tune grid to be put into the final model.
best_tunes <- select_best(tune_results)
#Including tuned specs from best tunes for the final model.
best_spec <- finalize_model(rand_forest_spec, best_tunes)
```

```
#set seed for tuned results
set.seed(123)
#set folds value for resampling
tree_folds <- vfold_cv(data = train_churn, v = 8, strata = churn)

#visualizing the tune results, to show what will be selected.

hyper_param_list <- read_rds("model_tuning.rds")
tune_results <- hyper_param_list$tune_results
best_spec <- hyper_param_list$best_spec

autoplot(tune_results)
```

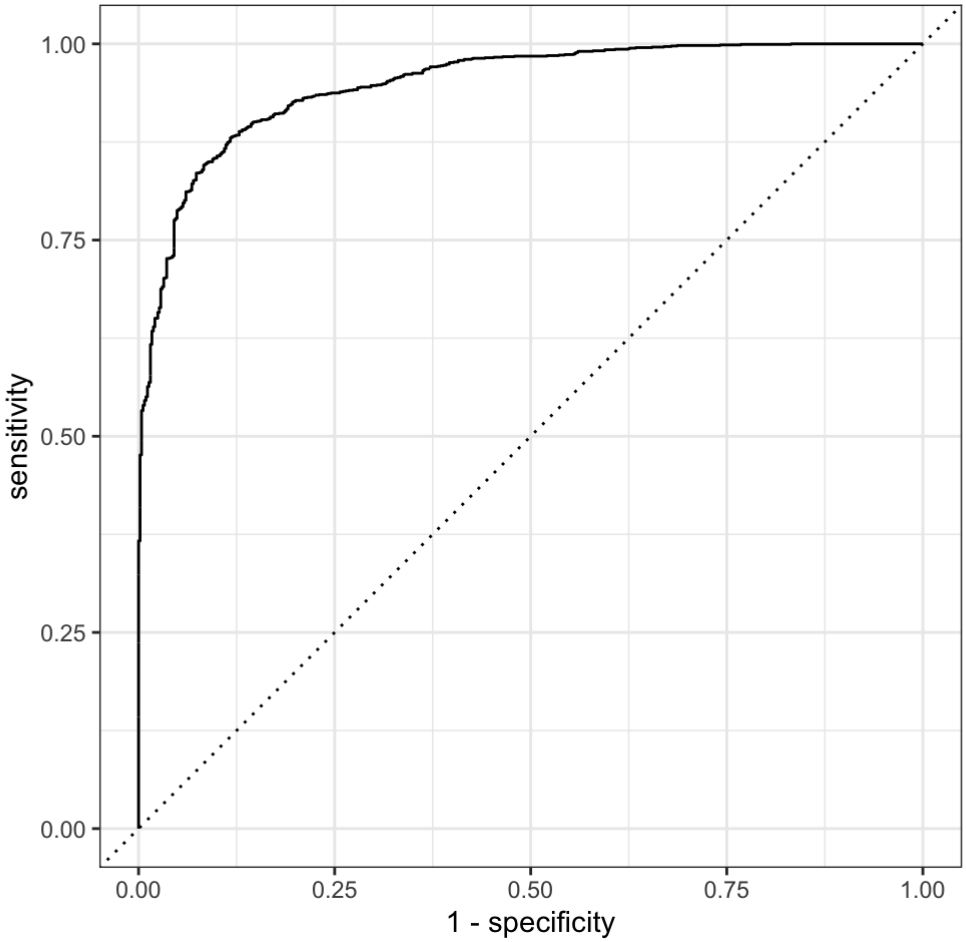


```
#finalizing the model
final_wflw <- workflow() %>%
  add_recipe(recipe_spec) %>%
  add_model(best_spec)

#add fit resample
fit_resample <- fit_resamples(
                           final_wflw,
                           resamples = tree_folds,
                           metrics   = metric_set(roc_auc, accuracy)
                     )

#using collect metrics to find the models performance
collect_metrics(fit_resample, summarize = TRUE)
```

```
## # A tibble: 2 × 6
##   .metric  .estimator  mean     n std_err .config
##   <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy binary     0.893     8 0.00330 Preprocessor1_Model1
## 2 roc_auc  binary     0.951     8 0.00228 Preprocessor1_Model1
```

D3. The following code is used to assess how the tuned model will perform on out of sample data.

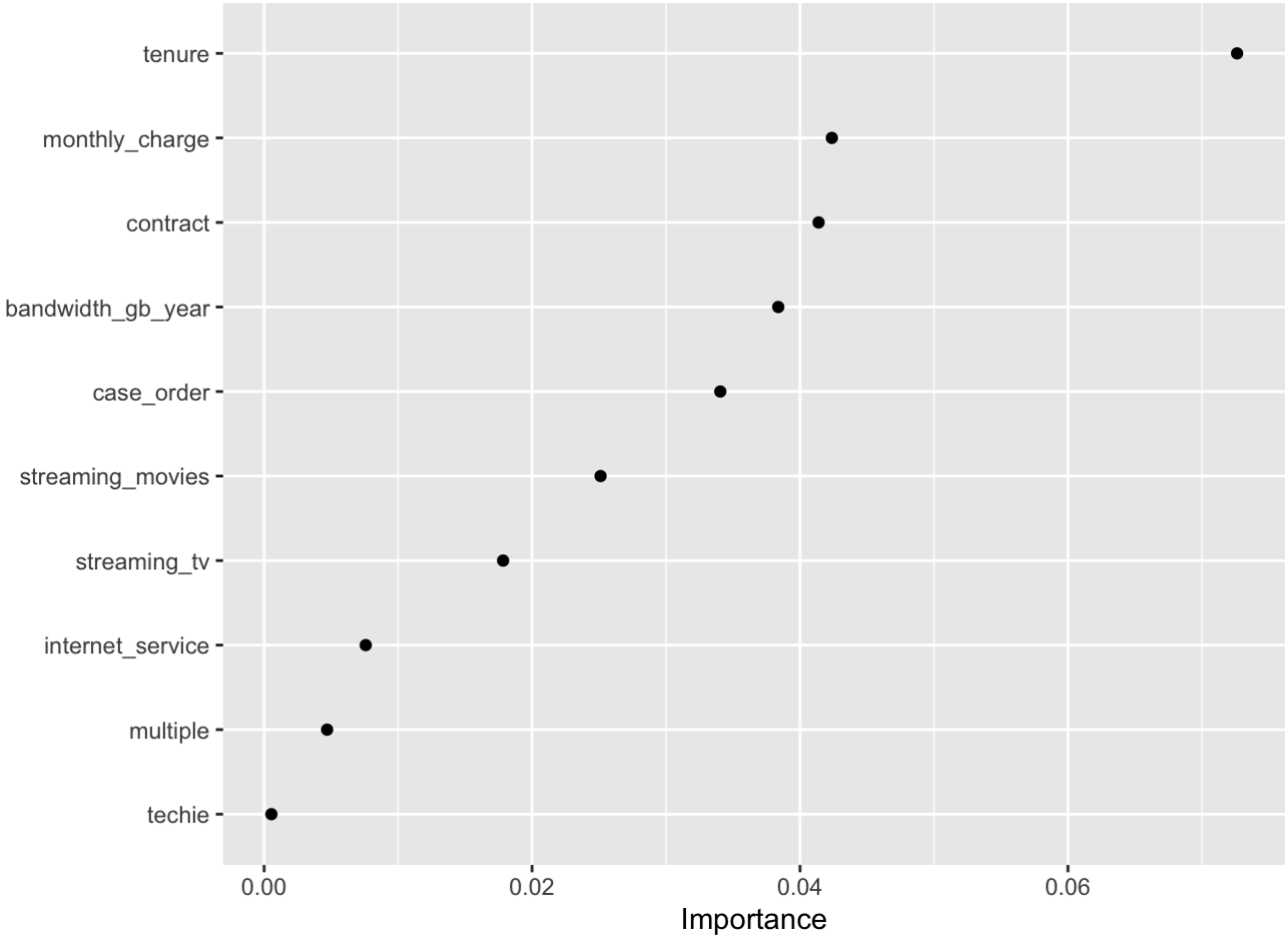
```
#full model created from work flow with all best tuning parameters
full_model <- final_wflw %>%
  last_fit(splits)

#visualizing the final full model.
full_model %>% collect_predictions() %>%
  roc_curve(truth = churn, estimate = .pred_No) %>%
  autoplot()
```




```
#viewing the metrics on the final full model.
full_model %>% collect_metrics()
```

```
## # A tibble: 2 × 4
##   .metric  .estimator .estimate .config
##   <chr>    <chr>          <dbl> <chr>
## 1 accuracy binary         0.883 Preprocessor1_Model1
## 2 roc_auc  binary         0.949 Preprocessor1_Model1
```

```
#Returns the workflow and display the most important variables from VIP package.
extract_workflow(full_model)%>%
  extract_fit_parsnip()%>%
  vip(geom="point",num_features=10)
```



E1. The prediction model has an accuracy value of 89.3% and a ROC AUC value of 95%. The accuracy of the model reveals that 10.7% of the observations are misclassified.

E2. From the full model using out of sample data, the model has an accuracy value of 88.3% and a ROC AUC value of 94.9%. This means that when given data that the model has not been learned from, the model will misclassify 11.4% of the observations.

E3. One limitation of the analysis technique is that the number of possible tuning parameter combinations that were used were limited by my machines capabilities. Since every combination of parameters needs to be tested by fitting the model to the values, computational time can quickly become very steep. The hyperparameter values were selected by applying best practice metrics from “Tidy Modeling with R” (Kuhn & Silge, 2022) and the R documentation for ranger random forest package.

E4.Overall the random forest model gave an improvement upon the KNN model from task 1. Stakeholders are still encouraged to seek out intervention opportunities that will help to mitigate churn rates. It is also advisable to monitor churn rates based on tenure, monthly charge, contract, bandwidth, case order, streaming movies, streaming tv, internet service and multiple service as these variables were selected by the random forest model as being the most important.

G. Citation

1. Machine Learning with Caret in R: https://app.datacamp.com/learn/courses/machine-learning-with-caret-in-r (https://app.datacamp.com/learn/courses/machine-learning-with-caret-in-r)

2. Machine Learning with Tree Models in R: https://app.datacamp.com/learn/courses/machine-learning-with-tree-based-models-in-r (https://app.datacamp.com/learn/courses/machine-learning-with-tree-based-models-in-r)

3. rand_forest: Random forest in parsnip: A Common API to Modeling and Analysis Functions Max Kuhn https://rdrr.io/cran/parsnip/man/rand_forest.html (https://rdrr.io/cran/parsnip/man/rand_forest.html)

4. Tidy Modeling with R. Silge. https://www.tmwr.org/ (https://www.tmwr.org/)

5. Applied Predictive Modeling. Kuhn. http://appliedpredictivemodeling.com/ (http://appliedpredictivemodeling.com/)