Project 2 Write Up

Kevin Puorro, Asitha Mudiyanselage, Caleb Skinner

Table of contents

Data Overview

There are 10,000 records in this data set, consisting of six continuous variables, two categorical variables, and three numeric binary variables.

There is no high correlation between any variables, and there are no missing values in this data set. The scales of the credit_score and estimated_salary differ from the other variables. This indicates that scaling the data set is necessary before applying a machine learning model. Below we display the basic summary statistics of the data set.

```
#   credit_score    geography           gender             age
#  Min.   :350.0   Length:10000       Length:10000       Min.   :18.00
#  1st Qu.:584.0   Class :character   Class :character   1st Qu.:32.00
#  Median :652.0   Mode  :character   Mode  :character   Median :37.00
#  Mean   :650.5                                         Mean   :38.92
#  3rd Qu.:718.0                                         3rd Qu.:44.00
#  Max.   :850.0                                         Max.   :92.00
#      tenure          balance       num_of_products has_cr_card is_active_member
#  Min.   : 0.000   Min.   :     0   Min.   :1.00    0:2945      0:4849
#  1st Qu.: 3.000   1st Qu.:     0   1st Qu.:1.00    1:7055      1:5151
#  Median : 5.000   Median : 97199   Median :1.00
#  Mean   : 5.013   Mean   : 76486   Mean   :1.53
#  3rd Qu.: 7.000   3rd Qu.:127644   3rd Qu.:2.00
#  Max.   :10.000   Max.   :250898   Max.   :4.00
#  estimated_salary    exited
#  Min.   :    11.58   0:7963
#  1st Qu.: 51002.11   1:2037
#  Median :100193.91
#  Mean   :100090.24
#  3rd Qu.:149388.25
#  Max.   :199992.48
```

Random Forest

Pick any machine learning method we have covered to predict ’exited‘ based on the other variables (except customer_id and surname). Be sure to do a training and testing split. Whatever method you choose to use, be sure to tune the model. Comment on the accuracy and confusion matrix for both the training set and the testing set.

We use a classification random forest model to predict whether a customer will exit the market. After standardizing the predictor variables, we use 100 trees and tune the mtry and min_n parameters. The tuning methods selected an mtry value of 5 and a minimum node of 50.

```
# # A tibble: 1 x 3
#    mtry min_n .config
#   <int> <int> <chr>
# 1     5    50 Preprocessor1_Model39
```

.metric	.estimator	.estimate
accuracy	binary	0.8815
kap	binary	0.5751

```
#           Truth
# Prediction    0    1
#          0 5814  731
#          1  158  796
```

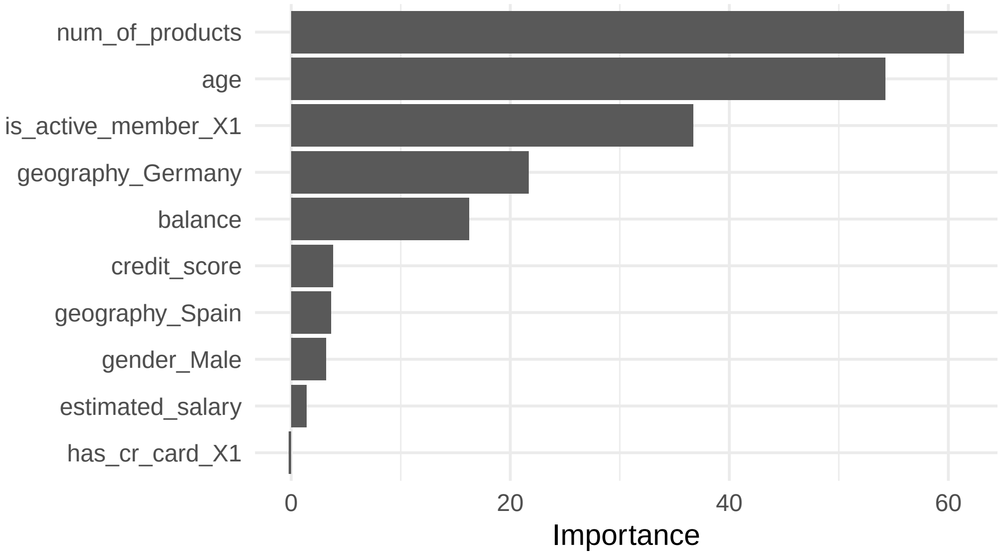
The model accurately classifies the exiting customer of the training set 88.1% of the time. The kappa estimate is 57.5%, showing that the model is not accurately predicting both outcomes at equal rates. The confusion matrix shows that most of the errors come when the model mislabels exiting customers as faithful ones. From the bank’s perspective, this is concerning.

.metric	.estimator	.estimate
accuracy	binary	0.8629
kap	binary	0.4982

```
#           Truth
# Prediction    0    1
#          0 1928  280
#          1   63  230
```

The model accurately classifies the exiting customer of the test set 86.3% of the time. The kappa estimate is 49.8%, showing that the model is not accurately predicting both outcomes at equal rates.

Lastly, we assessed each variable's importance in the final predictions. We found that the number of products and age are the most important variables in predicting if a customer will exit. Active membership, living in Germany, and balance are also very important predictors. The rest of the predictors seem to hold less overall value.



To ensure we found the most accurate model, we compare results from the Random Forest to that of several other models. Unsurprisingly, boosting (which is most similar to random forest) produces the second strongest accuracy. Logistic Regression performs decently well and the support vector machine is the least accurate.

method	accuracy
Random Forest	86.3%
Boosting	83.5%
Logistic Regression	82.1%
Support Vector Machine	79.6%

PCA Analysis

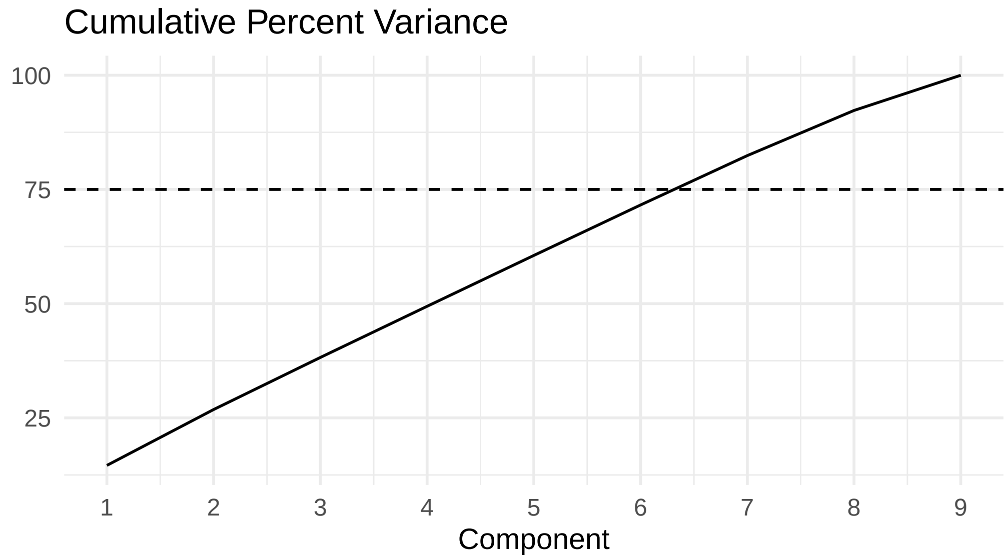
Use Principal Component Analysis to reduce the number of features (again, do not use customer_id or surname). Choose only the number of PCs that capture 75% of the variability.

We perform principal component analysis to reduce the number of features in the data set. The geography variable is categorical, and categorical variables cannot be represented well in PCA, so we remove it from our data set.

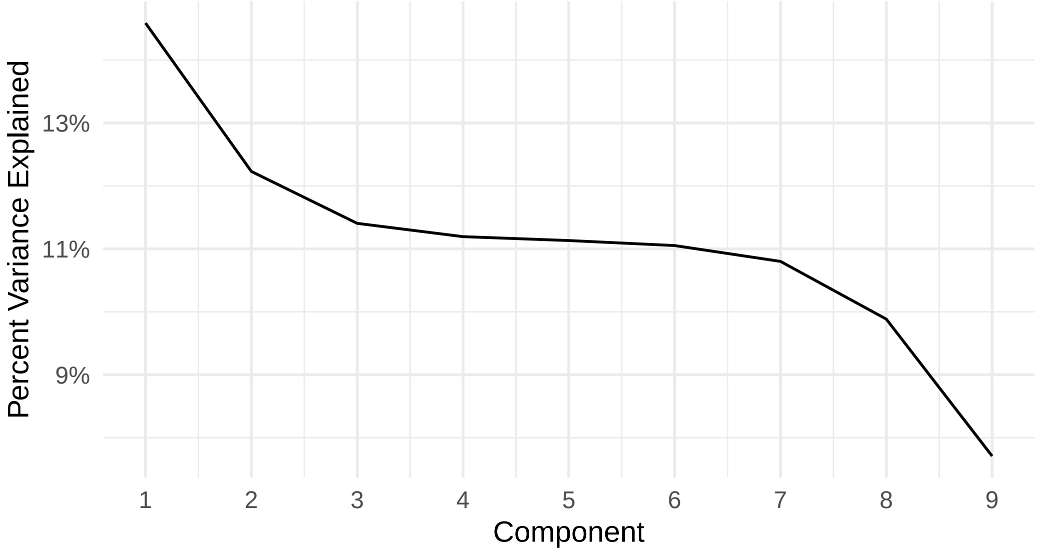
The "loadings" of each component signify how much each original variable contributes to the principal component. Hence, the balance and num_of_products have the highest loadings on PC1, indicating that they are the most influential variables in the variation captured by PC1. PC2 is dominated by the age and the is_active_member, and so forth.

```
# Importance of components:
#                           Comp.1    Comp.2    Comp.3    Comp.4    Comp.5
# Standard deviation     1.1458647 1.0491728 1.0131799 1.0037528 1.0009770
# Proportion of Variance 0.1458895 0.1223071 0.1140593 0.1119466 0.1113283
# Cumulative Proportion  0.1458895 0.2681966 0.3822559 0.4942025 0.6055308
#                           Comp.6    Comp.7     Comp.8     Comp.9
# Standard deviation     0.9973766 0.9859673 0.94317928 0.83291282
# Proportion of Variance 0.1105289 0.1080146 0.09884302 0.07708264
# Cumulative Proportion  0.7160597 0.8240743 0.92291736 1.00000000
#
# Loadings:
#                  Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8 Comp.9
# credit_score             0.164  0.116  0.654  0.249  0.659         0.180
# gender                          0.649  0.391 -0.211 -0.495         0.340
# age              -0.131  0.604        -0.333  0.310        -0.127  0.621
# tenure                  -0.296  0.327         0.629        -0.603 -0.179
# balance          -0.697                                                 -0.704
# num_of_products   0.698                                                 -0.706
# has_cr_card             -0.205  0.448 -0.448  0.268  0.274  0.638
# is_active_member         0.668  0.275               -0.161  0.122 -0.649
# estimated_salary        -0.115 -0.425  0.307  0.564 -0.449  0.434
```

Overall, the data is not reduced well by Principal Component Analysis. In order to explain 75% of the variance in the data, we need to include 7 of the 9 components. These 7 components explain 82.41% of the data.



The percent variance explained by each individual component is obviously highest for component 1, and it slightly decreases for the next few components. The third through seventh components explain roughly the same amount of data. Overall the first seven components are enough to explain 82% of the data. These seven components are then used in the random forest model in Part Three.



Random Forest of Principle Components

Redo the method you used in part 1 but this time use the PCs found in part 2 (only the PCs that account for 75% of the variability). Again, comment on the accuracy and confusion matrix for both the training and testing sets.

Now, we use our seven components to predict the customer exiting the market with a random forest model. We added geography back into the data set, because geography was removed to perform the principal component analysis. As with before, we use 75% of the data in a training set and 25% in a testing set. We also use five fold cross validation to tune the mtry and min_n parameters in the model. This random forest has 500 trees.

The tuning selected mtry = 6 and min_n = 45.

```
# # A tibble: 1 x 3
#    mtry min_n .config
#   <int> <int> <chr>
# 1     6    45 Preprocessor1_Model18
```

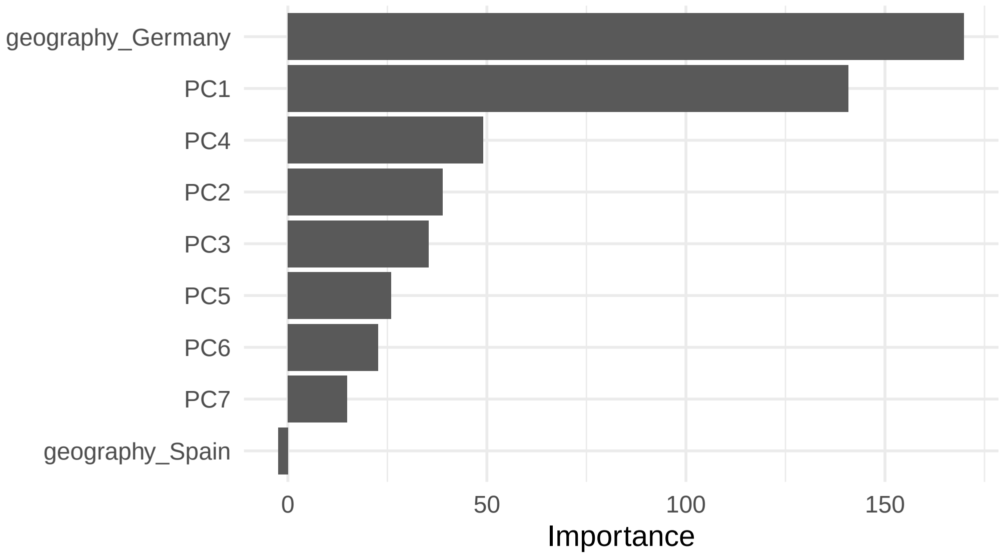
Overall, the model accurately classifies the exiting customer 82.45% of the time. The kappa estimate is 32.13%, showing that the model is not accurately predicting both outcomes at equal rates.

.metric	.estimator	.estimate
accuracy	binary	0.8245
kap	binary	0.3213

The findings of the confusion matrix fits with the kappa estimate. The model accurately classifies 28.6% of the customers who exit. Of the 231 customers the model classifies as exiters, 151 (68.01%) actually exit.

```
#           Truth
# Prediction    0    1
#          0 1911  359
#          1   80  151
```

We also looked at variable importance measurements for the model. This shows which variables are most significant in the classification of the customer. It's difficult to interpret the principal components, but it is interesting to see which components are most significant. The most important two variables are PC1 and living in Germany.


geography_Germany
PC1
PC4
PC2
PC3
PC5
PC6
PC7
geography_Spain
0
50
100
150
Importance

Appendix: code for all of this report

```r
# knitr defaults
knitr::opts_chunk$set(
  comment = "#", fig.height = 3,
  cache = FALSE, collapse = TRUE,
  error = TRUE, echo = FALSE,
  message = FALSE,
  warning = FALSE)

# libraries
library("tidyverse"); theme_set(theme_minimal())
library("tidymodels")
library("janitor")
library("xgboost")
library("vip")
library("flextable")

# set flextable defaults
set_flextable_defaults(
  font.size = 10, theme_fun = theme_apa,
  padding = 6,
  background.color = "#EFEFEF")

# create dataset
customer <- read_csv("CustomerChurn.csv") %>%
  clean_names() %>%
  select(-customer_id, -surname) %>%
  mutate(
    exited = factor(exited),
    has_cr_card = factor(has_cr_card),
    is_active_member = factor(is_active_member)) %>%
  na.omit()

# split data
set.seed(1128)
customer_split <- initial_split(customer, prop = .75, strata = exited)
customer_train <- training(customer_split)
customer_test <- testing(customer_split)

# cross validation folds
```

```
customer_folds <- vfold_cv(customer_train, v = 5, strata = exited)


#Let's look at the basic statistics of the data set

# Remove  CustomerId and Surname

summary(customer)

# glimpse(dat)

#Let's look at the  correlation matrix

# cust_r <- cor(dat[,-c(2,3,8,9)])

# corrplot(cust_r, method = "number",number.cex = 0.7)

# Count missing values

# colSums(is.na(customer))

# Frequency count of the target (Exited) variable

# count(customer, exited) %>%
#   flextable() %>%
#   align(align = "center")
# random forest model
rf_spec <- rand_forest(trees = 100,
                       mtry = tune(),
                       min_n = tune()) %>%
  set_engine("randomForest", importance = TRUE) %>%
  set_mode("classification")

rf_recipe <- recipe(exited ~ ., data = customer_train) %>%
  step_normalize(all_numeric_predictors()) %>%
  step_dummy(all_nominal_predictors())

#specify the tuning grid
rf_tuning_grid <- grid_regular(
  mtry(range = c(3, 6)),
  min_n(range = c(40, 50)),
  levels = 10
)

#fine tune the model
rf_tune_results <- tune_grid(
  object = workflow() %>%
    add_recipe(rf_recipe) %>%
    add_model(rf_spec),
  resamples = customer_folds,
  grid = rf_tuning_grid,
  metrics = metric_set(accuracy)
)

rf_best_params <- select_best(rf_tune_results, "accuracy")
rf_best_params

rf_fitted_model <- finalize_workflow(
  workflow() %>%
    add_recipe(rf_recipe) %>%
    add_model(rf_spec),
  rf_best_params) %>%
  fit(data = customer_train)

# train set performance
pred_train <- predict(rf_fitted_model, customer_train) %>%
  bind_cols(customer_train)

train_metrics <- metrics(pred_train, truth = exited, estimate = .pred_class)

train_conf_mat <- conf_mat(pred_train, truth = exited, estimate = .pred_class)

train_metrics %>% flextable() %>% align(align = "center") %>%
  colformat_double(j = 3, digits = 4)

print(train_conf_mat)

# test set performance
predictions <- predict(rf_fitted_model, customer_test) %>%
  bind_cols(customer_test)

test_metrics <- metrics(predictions, truth = exited, estimate = .pred_class)
```

```
test_conf_mat <- conf_mat(predictions, truth = exited, estimate = .pred_class)

test_metrics %>% flextable() %>% align(align = "center") %>%
  colformat_double(j = 3, digits = 4)

print(test_conf_mat)

# variable importance
vip(rf_fitted_model)

# table of other methods
tibble(
  method = c("Random Forest", "Boosting", "Logistic Regression", "Support Vector Machine")
  accuracy = c("86.3%", "83.5%", "82.1%", "79.6%")) %>%
  flextable() %>%
  align(align = "center", part = "all") %>%
  width(j = 1, width = 1.7)

# create pca df, converting binary variables into continuous
# remove geography because cannot be converted into ordinal values
customer2 <- customer %>%
  mutate(
    has_cr_card = if_else(has_cr_card == "1", 1, 0),
    is_active_member = if_else(is_active_member == "1", 1, 0),
    gender = if_else(gender == "Female", 0, 1),
    exited = if_else(exited == "1", 1, 0)) %>%
  select(-geography)

# PCA Analysis

set.seed(1124)

# dim(cust_dat)

# pca_df <- customer2[,-c(2,3,8,9)]

# dim(pca_df)

customer_pca <- customer2 %>% select(-exited) %>% princomp(cor =T)

summary(customer_pca, loadings = T)
```

```
# screeplot(customer_pca)
# conduct pca
pca_recipe <- recipe(exited ~ ., data = customer2) |>
              step_normalize(all_numeric_predictors()) |>
              step_pca(all_predictors(), threshold = .75)

# Prep the recipe to estimate PCA components

customer_prep_pca <- prep(pca_recipe, training = customer2)

# Extract the PCA results

customer_pca <- bake(customer_prep_pca, customer2) %>%
  mutate(
    exited = factor(exited),
    geography = customer$geography)

# View the results
tidy(customer_prep_pca, number = 2, type = "variance") %>%
  filter(terms == "cumulative percent variance") %>%
  ggplot() +
  geom_line(aes(x = component, y = value)) +
  scale_x_continuous(breaks = c(1,2,3,4,5,6,7,8,9)) +
  geom_hline(yintercept = 75, linetype = "dashed") +
  labs(x = "Component", title = "Cumulative Percent Variance", y = "")

# Plot Percent Variance of Components
tidy(customer_prep_pca, number = 2, type = "variance") %>%
  filter(terms == "percent variance") %>%
  mutate(value = value/100) %>%
  ggplot() +
  geom_line(aes(x = component, y = value)) +
  scale_x_continuous(breaks = c(1,2,3,4,5,6,7,8,9)) +
  labs(x = "Component", y = "Percent Variance Explained") +
  scale_y_continuous(labels = scales::label_percent())

# PCA Random Forest
set.seed(1128)
pca_split <- customer_pca %>% initial_split(prop = .75, strata = exited)

pca_train <- pca_split %>% training()
```

```r
pca_test <- pca_split %>% testing()
pca_folds <- vfold_cv(pca_train, v = 5)

rf_spec <- rand_forest(trees = 500,
                       mtry = tune(),
                       min_n = tune()) %>%
  set_engine("randomForest", importance = TRUE) %>%
  set_mode("classification")

rf_pca_rec <- recipe(exited ~ ., data = pca_train) %>%
  step_normalize(all_numeric_predictors()) %>%
  step_dummy(all_nominal_predictors())

rf_tuning_grid <- grid_regular(
  mtry(range = c(4, 6)),
  min_n(range = c(40, 50)),
  levels = 10)

rf_pca_tune_results <- tune_grid(
  object = workflow() %>%
    add_recipe(rf_pca_rec) %>%
    add_model(rf_spec),
  resamples = pca_folds,
  grid = rf_tuning_grid,
  metrics = metric_set(accuracy))

rf_pca_best_params <- select_best(rf_pca_tune_results, "accuracy")
rf_pca_best_params

rf_pca_final <- finalize_workflow(
  workflow() %>%
    add_recipe(rf_pca_rec) %>%
    add_model(rf_spec),
  rf_pca_best_params) %>%
  fit(data = pca_train)

predictions <- augment(rf_pca_final, new_data = pca_test)

# accuracy
metrics(predictions, truth = exited, estimate = .pred_class) %>%
  flextable() %>%
  align(align = "center") %>%
  colformat_double(j = 3, digits = 4)

# confusion matrix
conf_mat(predictions, truth = exited, estimate = .pred_class)

# variable importance
vip(rf_pca_final)
```