Tackling Attrition at Tifosi Bank_v2

```
#package
library(tidyverse)
library(tidymodels)
library(janitor)
library(skimr)
library(here)
library(readr)
tidymodels_prefer()
```

```
raw_data<-read_csv(here("Data","bank_churners.csv")) %>%
  clean_names()
```

data split

```
set.seed(47969938)
data_split2<- initial_split(raw_data, prop = 0.8, strata = attrition_flag)
train_data2<- training(data_split2)
test_data2 <- testing(data_split2)
# Cross-validation folds
cross_validation_folds2 <- vfold_cv(train_data2, v = 10, strata = attrition_flag)
```

Logistic base model - clean version

```
# Define model
base_logistic_spec2 <- logistic_reg() %>%
  set_engine("glm") %>%
```

```
  set_mode("classification")

# Define recipe ----------------------------------------------------------------
logistic_recipe2 <- recipe(attrition_flag ~ ., data = train_data2) |>
  step_other(all_nominal_predictors(), threshold = 0.05) |>
  step_dummy(all_nominal_predictors()) |>
  step_zv(all_predictors()) |>
  step_corr(threshold = 0.7) |>
  step_normalize(all_numeric_predictors())

# Create workflow --------------------------------------------------------------
base_logistic_wf2 <- workflow() |>
  add_model(base_logistic_spec2) |>
  add_recipe(logistic_recipe2)

# Cross-validation: fit_resamples ----------------------------------------------
base_logistic_cv2 <- fit_resamples(
  base_logistic_wf2,
  resamples = cross_validation_folds2,
  metrics = metric_set(roc_auc, accuracy)
)

# Evaluation on cross-validation folds -----------------------------------------
collect_metrics(base_logistic_cv2)
```

```
# A tibble: 2 x 6
  .metric  .estimator  mean     n std_err .config
  <chr>    <chr>      <dbl> <int>   <dbl> <chr>
1 accuracy binary     0.903    10 0.00365 Preprocessor1_Model1
2 roc_auc  binary     0.923    10 0.00279 Preprocessor1_Model1
```

```
# Last fit:  train_data2 fit  predict test_data2
final_base_fit2 <- last_fit(
  base_logistic_wf2,
  split = data_split2,
  metrics = metric_set(roc_auc, accuracy)
)

# Collect performance on test set
collect_metrics(final_base_fit2)
```

```
# A tibble: 2 x 4
```

```
  .metric  .estimator .estimate .config
  <chr>    <chr>          <dbl> <chr>
1 accuracy binary         0.909 Preprocessor1_Model1
2 roc_auc  binary         0.931 Preprocessor1_Model1
```

Ridge Logistic Regression (mixture = 0) - clean version

```
# Ridge Logistic Regression (mixture = 0) - clean version

# Define model
ridge_spec2 <- logistic_reg(
  penalty = tune(),    # penalty tune
  mixture = 0          # mixture = 0 = Ridge
) %>%
  set_engine("glmnet") %>%
  set_mode("classification")

# Create workflow
ridge_wf2 <- workflow() %>%
  add_model(ridge_spec2) %>%
  add_recipe(logistic_recipe2)   # <---   recipe2

# Define tuning grid
ridge_grid2 <- grid_regular(
  penalty(range = c(-4, 0)),
  levels = 30
)

# Tuning
ridge_tune2 <- tune_grid(
  ridge_wf2,
  resamples = cross_validation_folds2,  # <---   folds2
  grid = ridge_grid2,
  metrics = metric_set(roc_auc, accuracy)
)
```

```
Warning: package 'glmnet' was built under R version 4.3.3
```

```
Warning: package 'Matrix' was built under R version 4.3.3
```

```r
# Select best model
best_ridge2 <- select_best(ridge_tune2, metric = "roc_auc")

# Finalize workflow
final_ridge_wf2 <- finalize_workflow(ridge_wf2, best_ridge2)

# Last fit
final_ridge_fit2 <- last_fit(
  final_ridge_wf2,
  split = data_split2,   # <---   split2
  metrics = metric_set(roc_auc, accuracy)
)

# Collect metrics
collect_metrics(final_ridge_fit2)
```

```
# A tibble: 2 x 4
  .metric  .estimator .estimate .config
  <chr>    <chr>          <dbl> <chr>
1 accuracy binary         0.905 Preprocessor1_Model1
2 roc_auc  binary         0.922 Preprocessor1_Model1
```

Lasso Logistic Regression (mixture = 1) - clean version

```r
# Lasso Logistic Regression (mixture = 1) - clean version

# Define model
lasso_spec2 <- logistic_reg(
  penalty = tune(),   # penalty tune
  mixture = 1         # mixture = 1 = Lasso
) %>%
  set_engine("glmnet") %>%
  set_mode("classification")

# Create workflow
lasso_wf2 <- workflow() %>%
  add_model(lasso_spec2) %>%
  add_recipe(logistic_recipe2)  #   recipe2
```

```r
# Define tuning grid
lasso_grid2 <- grid_regular(
  penalty(range = c(-4, 0)),
  levels = 30
)

# Tuning
lasso_tune2 <- tune_grid(
  lasso_wf2,
  resamples = cross_validation_folds2,  #    folds2
  grid = lasso_grid2,
  metrics = metric_set(roc_auc, accuracy)
)

# Select best model
best_lasso2 <- select_best(lasso_tune2, metric = "roc_auc")

# Finalize workflow
final_lasso_wf2 <- finalize_workflow(lasso_wf2, best_lasso2)

# Last fit
final_lasso_fit2 <- last_fit(
  final_lasso_wf2,
  split = data_split2,   #    split2
  metrics = metric_set(roc_auc, accuracy)
)

# Collect metrics
collect_metrics(final_lasso_fit2)
```

```
# A tibble: 2 x 4
  .metric  .estimator .estimate .config
  <chr>    <chr>          <dbl> <chr>
1 accuracy binary         0.909 Preprocessor1_Model1
2 roc_auc  binary         0.931 Preprocessor1_Model1
```

Elastic Net Logistic Regression - clean version

```r
# Elastic Net Logistic Regression - clean version

# Define model
elastic_spec2 <- logistic_reg(
  penalty = tune(),
  mixture = tune()    # Elastic Net tune mixture
) %>%
  set_engine("glmnet") %>%
  set_mode("classification")

# Create workflow
elastic_wf2 <- workflow() %>%
  add_model(elastic_spec2) %>%
  add_recipe(logistic_recipe2)  #   recipe2

# Define tuning grid
elastic_grid2 <- grid_regular(
  penalty(range = c(-4, 0)),
  mixture(range = c(0, 1)),
  levels = c(20, 5)             # penalty 20 mixture 5
)

# Tuning
elastic_tune2 <- tune_grid(
  elastic_wf2,
  resamples = cross_validation_folds2,  #    folds2
  grid = elastic_grid2,
  metrics = metric_set(roc_auc, accuracy)
)

# Select best model
best_elastic2 <- select_best(elastic_tune2, metric = "roc_auc")

# Finalize workflow
final_elastic_wf2 <- finalize_workflow(elastic_wf2, best_elastic2)

# Last fit
final_elastic_fit2 <- last_fit(
  final_elastic_wf2,
  split = data_split2,  #    split2
  metrics = metric_set(roc_auc, accuracy)
)
```

```r
# Collect metrics
collect_metrics(final_elastic_fit2)
```

```
# A tibble: 2 x 4
  .metric  .estimator .estimate .config
  <chr>    <chr>          <dbl> <chr>
1 accuracy binary         0.909 Preprocessor1_Model1
2 roc_auc  binary         0.931 Preprocessor1_Model1
```

KNN - clean version

```r
# KNN - clean version

# 1.1 Define KNN model
knn_spec2 <- nearest_neighbor(
  neighbors = tune()
) %>%
  set_engine("kknn") %>%
  set_mode("classification")

# 1.2 Define Recipe  normalize
knn_recipe2 <- recipe(attrition_flag ~ ., data = train_data2) %>%
  step_other(all_nominal_predictors(), threshold = 0.05) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_zv(all_predictors()) %>%
  step_corr(threshold = 0.7) %>%
  step_normalize(all_numeric_predictors())  # KNN  normalize

# 1.3 Create Workflow
knn_wf2 <- workflow() %>%
  add_model(knn_spec2) %>%
  add_recipe(knn_recipe2)

# 1.4 Define Grid
knn_grid2 <- grid_regular(
  neighbors(range = c(3, 50)),
  levels = 10
)
```

```
# 1.5 Tune
knn_tuned2 <- tune_grid(
  knn_wf2,
  resamples = cross_validation_folds2,   #    folds2
  grid = knn_grid2,
  metrics = metric_set(roc_auc, accuracy)
)
```

```
Warning: package 'kknn' was built under R version 4.3.3
```

```
# 1.6 Select best
knn_best2 <- select_best(knn_tuned2, metric = "roc_auc")

# 1.7 Finalize
final_knn_wf2 <- finalize_workflow(knn_wf2, knn_best2)

# 1.8 Last Fit
knn_final_fit2 <- last_fit(
  final_knn_wf2,
  split = data_split2  #    split2
)

# 1.9 Collect metrics
collect_metrics(knn_final_fit2)
```

```
# A tibble: 3 x 4
  .metric     .estimator .estimate .config
  <chr>       <chr>          <dbl> <chr>
1 accuracy    binary        0.868  Preprocessor1_Model1
2 roc_auc     binary        0.882  Preprocessor1_Model1
3 brier_class binary        0.0927 Preprocessor1_Model1
```

Random Forest - clean version

```
# Random Forest - clean version

# 2.1 Define Random Forest model
rf_spec2 <- rand_forest(
  mtry = tune(),
```

```
  min_n = tune()
) %>%
  set_engine("ranger") %>%
  set_mode("classification")

# 2.2 Define Recipe RF  normalize
rf_recipe2 <- recipe(attrition_flag ~ ., data = train_data2) %>%
  step_other(all_nominal_predictors(), threshold = 0.05) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_zv(all_predictors()) %>%
  step_corr(threshold = 0.7)

# 2.3 Create Workflow
rf_wf2 <- workflow() %>%
  add_model(rf_spec2) %>%
  add_recipe(rf_recipe2)

# 2.4 Define Grid
rf_grid2 <- grid_regular(
  mtry(range = c(2, 10)),
  min_n(range = c(5, 30)),
  levels = 5
)

# 2.5 Tune
rf_tuned2 <- tune_grid(
  rf_wf2,
  resamples = cross_validation_folds2,  #    folds2
  grid = rf_grid2,
  metrics = metric_set(roc_auc, accuracy)
)
```

```
Warning: package 'ranger' was built under R version 4.3.3
```

```
# 2.6 Select best
rf_best2 <- select_best(rf_tuned2, metric = "roc_auc")

# 2.7 Finalize
final_rf_wf2 <- finalize_workflow(rf_wf2, rf_best2)

# Last fit
final_rf_fit2 <- last_fit(
```

```
  final_rf_wf2,
  split = data_split2,  #    split2
  metrics = metric_set(roc_auc, accuracy)
)

# Collect metrics
collect_metrics(final_rf_fit2)
```

```
# A tibble: 2 x 4
  .metric  .estimator .estimate .config
  <chr>    <chr>          <dbl> <chr>
1 accuracy binary         0.966 Preprocessor1_Model1
2 roc_auc  binary         0.993 Preprocessor1_Model1
```

XGBoost - clean version

```
# 3.1 Define XGBoost model
xgb_spec2 <- boost_tree(
  trees = tune(),
  tree_depth = tune(),
  learn_rate = tune(),
  loss_reduction = tune(),
  sample_size = tune(),
  mtry = tune()
) %>%
  set_engine("xgboost") %>%
  set_mode("classification")

# 3.2 Define Recipe XGB   normalize
xgb_recipe2 <- recipe(attrition_flag ~ ., data = train_data2) %>%
  step_other(all_nominal_predictors(), threshold = 0.05) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_zv(all_predictors()) %>%
  step_corr(threshold = 0.7)

# 3.3 Create Workflow
xgb_wf2 <- workflow() %>%
  add_model(xgb_spec2) %>%
  add_recipe(xgb_recipe2)
```

```r
# 3.4 Define Grid
xgb_grid2 <- grid_latin_hypercube(
  trees(range = c(100, 1000)),
  tree_depth(range = c(2, 10)),
  learn_rate(range = c(0.01, 0.3)),
  loss_reduction(range = c(0.0001, 1)),
  sample_size = sample_prop(range = c(0.5, 1)),
  mtry(range = c(2, 10)),
  size = 20
)
```

```
Warning: `grid_latin_hypercube()` was deprecated in dials 1.3.0.
i Please use `grid_space_filling()` instead.
```

```r
# 3.5 Tune
xgb_tuned2 <- tune_grid(
  xgb_wf2,
  resamples = cross_validation_folds2,  #    folds2
  grid = xgb_grid2,
  metrics = metric_set(roc_auc, accuracy)
)
```

```
Warning: package 'xgboost' was built under R version 4.3.3
```

```r
# 3.6 Select best
xgb_best2 <- select_best(xgb_tuned2, metric = "roc_auc")

# 3.7 Finalize
final_xgb_wf2 <- finalize_workflow(xgb_wf2, xgb_best2)

# Last fit
final_xgb_fit2 <- last_fit(
  final_xgb_wf2,
  split = data_split2,  #    split2
  metrics = metric_set(roc_auc, accuracy)
)

# Collect metrics
collect_metrics(final_xgb_fit2)
```

```
# A tibble: 2 x 4
  .metric  .estimator .estimate .config
  <chr>    <chr>          <dbl> <chr>
1 accuracy binary         0.963 Preprocessor1_Model1
2 roc_auc  binary         0.991 Preprocessor1_Model1
```