```
# ------- To be used to  model occurrence  by applying  multiple  class balancing approaches
-------
# -------------- load libraries to be used -------------------------------
libs <- c('yardstick','parsnip','rsample','recipes','tidymodels','tune','tidyverse','doParallel',
          'ggridges','ggrepel','randomForest','themis','janitor');
lapply(libs,library,character.only = TRUE)

# ------------- get the data to be used  ---------------------------------
# ------  load the data to be used ----------------------------------------
occ_data =
read_csv('SPECIFY_FOLDER_WHERE_DATA_FILE_IS_LOCATED/all_whale_enviro_data.
csv')

#--- Plot timeseries of environmental conditions ----#
occ_data$date=format(as.Date(occ_data$datetime), "%d/%B/%Y")

occ_data_vers<-occ_data%>%
              pivot_longer(names_to='species',values_to = 'values', cols=c('Dcalls':'Upcalls'))


#--- Set data for modeling ----#
# -------- retain only species with occurrence >=20  ----------------------#
sel_species =
occ_data_vers%>%filter(values==1)%>%dplyr::select(species)%>%distinct()%>%pull()

# ------  Multicollinearity test ----------------------------------------
## All other whale species ##
car::vif(lm(Month ~ chl+Hour+sst_nrt+windsp+ssh,data = occ_data))

## Antarctic blue whales ##
car::vif(lm(Month ~ chl_abw+Hour+sstC_abw+windsp_abw+ssh_abw,data = occ_data))

## Madagascan pygmy blue whales ##
car::vif(lm(Month ~ chl_pbw+Hour+sstC_pbw+windsp_pbw+ssh_pbw,data = occ_data))

# ------  prepare data to be used ----------------------------------------
sel_var_all=c("sst_nrt","sstC_abw","sstC_pbw", "ssh","ssh_abw","ssh_pbw", "windsp",
"windsp_abw","windsp_pbw", "Hour",
             "Month", "chl","chl_abw", "chl_pbw")

occ_sel  = occ_data%>%
  dplyr::select(all_of(c(sel_var_all,sel_species)))%>%
  mutate_at(.vars = vars(c("Madblues", "Antblues","Pulse20Hz", "Dcalls", "Pulses40Hz",
"Upcalls",  "Humpbacks",
                           "Minkes")),.funs = as.factor)


# ----------  set formula and recipe -----------------------------------
# ---------- model to be fitted ----------------------------------------

forms_baleen <- tibble(models=paste('model',1:8,sep = '_'),
                    type = c("Madblues", "Antblues","Pulse20Hz", "Dcalls", "Pulses40Hz",
"Upcalls",  "Humpbacks",
                             "Minkes"),
```

```
                forms =
c("Madblues~Month+Hour+sstC_pbw+chl_pbw+ssh_pbw+windsp_pbw",
                  "Antblues~Month+Hour+sstC_abw+chl_abw+ssh_abw+windsp_abw",
                  "Pulse20Hz~Month+Hour+sstC_abw+chl_abw+ssh_abw+windsp_abw",
                  "Dcalls~Month+Hour+sst_nrt+chl+ssh+windsp",
                  "Pulses40Hz~Month+Hour+sst_nrt+chl+ssh+windsp",
                  "Upcalls~Month+Hour+sst_nrt+chl+ssh+windsp",
                  "Humpbacks~Month+Hour+sst_nrt+chl+ssh+windsp",
                  "Minkes~Month+Hour+sst_nrt+chl+ssh+windsp"),
              raw_dat = list(occ_sel%>%dplyr::select(all_of(sel_var_all),Madblues),
                  occ_sel%>%dplyr::select(all_of(sel_var_all),Antblues),
                  occ_sel%>%dplyr::select(all_of(sel_var_all),Pulse20Hz),
                  occ_sel%>%dplyr::select(all_of(sel_var_all),Dcalls),
                  occ_sel%>%dplyr::select(all_of(sel_var_all),Pulses40Hz),
                  occ_sel%>%dplyr::select(all_of(sel_var_all),Upcalls),
                  occ_sel%>%dplyr::select(all_of(sel_var_all),Humpbacks),
                  occ_sel%>%dplyr::select(all_of(sel_var_all),Minkes)))

# -------------- apply class balancing algorithm -------------------------
prep_balancing = function(the_recipe,the_data,pred_var){
  cust_switch = function(recipes,the_pred,type){
    if(type=='un_balanced'){
      samp_rec = recipes
    }else if(type=='up_sampled'){
      samp_rec = recipes%>%themis::step_upsample(the_pred,seed = 1234)
    }else if(type=='down_sampled'){
      samp_rec = recipes%>%themis::step_downsample(the_pred,seed = 1234)
    }else if(type=='smote'){
      samp_rec = recipes%>%themis::step_smote(the_pred,seed = 1234)
    }else if(type=='adasyn'){
      samp_rec = recipes%>%themis::step_adasyn(the_pred,seed = 1234)
    }
    samp_rec
  }

balanc_dat = tibble(balancing =
c('un_balanced','up_sampled','down_sample','smote','adasyn'),
                    balancing_dat = list(the_recipe%>%prep(.)%>%bake(new_data = NULL),
                                         the_recipe%>%themis::step_upsample(pred_var,seed =
124)%>%
                                           prep()%>%bake(new_data = NULL),
                                         the_recipe%>%themis::step_downsample(pred_var,seed =
124)%>%
                                           prep()%>%bake(new_data = NULL),
                                         the_recipe%>%themis::step_smote(pred_var,seed = 124)%>%
                                           prep()%>%bake(new_data = NULL),
                                         the_recipe%>%themis::step_adasyn(pred_var,seed = 124)%>%
                                           prep()%>%bake(new_data = NULL)))

  balanc_dat

}


forms_baleen = forms_baleen%>%
```

```
  mutate(recipes = purrr::map2(.x = forms,.y = raw_dat,.f = ~recipe(formula(.x),data = .y)),
         mult_balance = purrr::pmap(.l = list(the_recipe = recipes,the_data = raw_dat,pred_var
= as.list(type)),
                                    .f = prep_balancing))

# ----------- prep data for tuning ---------------------------------------
cust_init_split = function(the_data){set.seed(1245);initial_split(the_data,prop = .7)}
cust_cv_train = function(the_train){set.seed(12345);vfold_cv(data = the_train,v = 10,repeats
= 3)}

prep_train_baleen = forms_baleen%>%dplyr::select(-raw_dat)%>%
  unnest(cols = mult_balance)

prep_train_baleen = prep_train_baleen%>%
  mutate(ini_split = purrr::map(.x = balancing_dat,.f = ~cust_init_split(the_data = .x)),
         train_dat = purrr::map(.x = ini_split,.f = ~training(.x)),
         cv_dat = purrr::map(.x = train_dat,.f = ~cust_cv_train(the_train = .x)))

# ---------- tune and test the model ----------------------------------------
source('SPECIFY_FOLDER_WHERE_SOURCE_FILE_IS_LOCATED/000_source_all.R')

cust_rf_parallel = function(recipes_al,trains_al,splits_al){
  ncl <-  makePSOCKcluster(5)
  registerDoParallel(ncl)
  rf_rslt = cust_rf_tune_test_class(the_recipe = recipes_al,
                        train_rsample = trains_al,
                        final_split =splits_al)
  stopCluster(ncl)
  rf_rslt
}


system.time(prep_train_baleen <- prep_train_baleen%>%
              mutate(model_out = purrr::pmap(.l = list(recipes_al = recipes,trains_al = cv_dat,
                                                       splits_al = ini_split),.f = cust_rf_parallel)))

#  ------ write  model tuning and test outputs to file ------------------
save(list = c('prep_train_baleen'),
     file =
'SPECIFY_FOLDER_TO_SAVE_FILE/PEIs_tune_test_baleen_whales_occ_balance.RData')
```