```
# --------- To be used to process model output: balanced occurrence modeling----------------------------
# ---------------------- load libraries ---------------------------------
libs <- c('yardstick','parsnip','rsample','recipes','tidymodels','tune','tidyverse','doParallel',
          'vip','ranger','randomForest','ppsr', 'janitor');
purrr::map(libs,~library(.x,character.only=TRUE))

# -------- load tuning output ---------------------------------------------
load('SPECIFY_FOLDER_WHERE_DATA_FILE_IS_LOCATED/PEIs_tune_test_baleen_whales_occ_
balance.RData')

species_map = tibble(spp_code = c("Madblues", "Antblues","Pulse20Hz", "Dcalls", "Pulses40Hz",
"Upcalls",  "Humpbacks",
                                  "Minkes"),
                     species = c('MPBW calls',"ABW Z-calls", "Fin whale 20 Hz pulses", "Blue whale D-calls",
"Fin whale 40 Hz pulses",
                                 "Sei whale upsweep calls", "Humpback whale songs", "AMW bioduck calls"))


# ---------  extract optimal tuning parameters ----------------------------
all_opt_tune_occ =
bind_rows(prep_train_baleen%>%dplyr::select(models,forms,type,balancing,model_out)%>%
                            unnest(cols = model_out)%>%
                            dplyr::select(models,forms,type,balancing,opt_tune)%>%unnest(cols = opt_tune))

# -------  process tuning and test results -------------------------------
all_test_perf_occ =
bind_rows(prep_train_baleen%>%dplyr::select(models,type,balancing,model_out)%>%
                             unnest(cols = model_out)%>%
                             dplyr::select(models,type,balancing,test_perf)%>%unnest(cols =test_perf)%>%

dplyr::filter(.metric=='roc_auc')%>%dplyr::select(models,type,balancing,.estimate))

all_test_perf_occ = all_test_perf_occ%>%
  left_join(species_map,by = c('type' = 'spp_code'))%>%
  mutate(balancing = forcats::fct_relevel(balancing,c("un_balanced", "down_sample","up_sampled",
"smote", "adasyn"
  )))

# -------  visual summary test performance  ------------------------------
source('SPECIFY_FOLDER_WHERE_DATA_FILE_IS_LOCATED/000_source_all.R')

cols_fill = c("antiquewhite4", "blue", "blueviolet", "darkred", "darkgreen", "coral", "lightsteelblue")

all_test_perf_occ<-all_test_perf_occ%>%
  mutate(balancingF = case_when(
    balancing=="un_balanced" ~ "Unbalanced",
    balancing=="down_sample" ~ "Downsampling",
    balancing=="up_sampled" ~ "Upsampling",
    balancing=="smote" ~ "SMOTE",
    balancing=="adasyn" ~ "ADASYN"))

all_test_perf_occ_f<- all_test_perf_occ%>%
  mutate(balancing_n = case_when(
    balancingF=="Unbalanced"~4,
    balancingF=="Downsampling"~2,
    balancingF=="Upsampling"~5,
    balancingF=="SMOTE"~3,
    balancingF=="ADASYN"~1))

all_test_perf_occ_f$alTInd = all_test_perf_occ_f$balancing_n/5
all_test_perf_occ_f$balancingF=reorder(all_test_perf_occ_f$balancingF,all_test_perf_occ_f$alTInd,m
ean)

model_per_mtrics_baleen = all_test_perf_occ_f%>%
  ggplot()+
  geom_bar(mapping = aes(x = type,y = .estimate,fill = balancingF),
```

```
          stat = 'identity',position = position_dodge(),alpha = 0.6)+
  scale_fill_manual(values = cols_fill[1:5],name = 'class balancing')+
  labs(y = 'AUC')+
  geom_hline(yintercept = 0.7,linetype=2)+
  #facet_wrap(~species)+
  theme_bw()+custom_ggplot_themes(legend_position = 'top')#theme(strip.background =
element_rect(fill = 'white'))

model_per_mtrics_baleen

ggsave(filename = 'SPECIFY_FOLDER_SAVE_FILE/model_performance_by_class_balancing.png',
       plot = model_per_mtrics_baleen,device = 'png',width = 30,height = 27,units = 'cm',dpi = 200)

# ------------ calibration plot -----------------------------------------
all_calib_data =
bind_rows(prep_train_baleen%>%dplyr::select(models,type,balancing,model_out)%>%
                        unnest(cols = model_out)%>%
                        dplyr::select(models,type,balancing,test_pred)%>%unnest(cols =test_pred)%>%
                        dplyr::select(-c(.pred_0,.row,.pred_class,.config))%>%
                        pivot_longer(cols = c('Madblues':'Minkes'),names_to = 'spp_code',values_to =
'values')%>%
                        filter(!is.na(values)))

all_calib_data = all_calib_data%>%
  left_join(species_map)%>%
  nest(data = c(.pred_1,species,values))%>%
  left_join(species_map)%>%
  dplyr::select(-spp_code)

# -------- generate  data for calibration and plot calibration ------------

all_calib_data<-all_calib_data%>%
  mutate(balancingF = case_when(
    balancing=="un_balanced" ~ "Unbalanced",
    balancing=="down_sample" ~ "Downsampling",
    balancing=="up_sampled" ~ "Upsampling",
    balancing=="smote" ~ "SMOTE",
    balancing=="adasyn" ~ "ADASYN"))

all_calibs = all_calib_data%>%
  mutate(calib = purrr::map(.x = data,.f = ~calib_plot_vals(the_data = .x,species_name = species)),
         calib_plot = purrr::map2(.x = calib,.y = as.list(balancingF),.f = ~gg_calib_plot(dat = .x,main_title
= .y)))

# Calibration plot

calib_baleen_plt = all_calibs%>%
  unnest(calib)%>%
  ggplot(aes(x = bin_pred, y = bin_prob, ymin = ll, ymax = ul)) +
  geom_pointrange(size = 0.5, color = "black") +
  scale_y_continuous(limits = c(0, 1), breaks = seq(0, .9, by = 0.2)) +
  scale_x_continuous(limits = c(0, 1), breaks = seq(0, .9, by = 0.2)) +
  geom_abline() + # 45 degree line indicating perfect calibration
  labs(x = 'Predicted probability',y = "Observed probability",subtitle = '')+
  geom_smooth(data = all_calibs%>%
                unnest(calib),method = "lm", se = FALSE, linetype = "dashed",
              color = "black", formula = y~-1 + x) +
  # straight line fit through estimates
  geom_smooth(data = all_calibs%>%
                unnest(calib),aes(x = .pred_1, y = as.numeric(as.character(values))),
              color = "red", se = FALSE, method = "loess") +
  facet_grid(balancingF ~ species)+
  # loess fit through estimates

  theme_bw()+#custom_ggplot_themes()
  theme(strip.background = element_rect(fill = 'white'),
```

```
      strip.text = element_text(size = 17),
      text = element_text(size=20), axis.title = element_text(size=22),
      axis.text=element_text(size=20),axis.text.x=element_text(size=18),
      panel.grid.major = element_blank(), panel.grid.minor = element_blank())

calib_baleen_plt

ggsave(filename = 'SPECIFY_FOLDER_SAVE_FILE/calibration_plot_by_class_balancing.png',
       plot = calib_baleen_plt,device = 'png',width = 60,height = 35,units = 'cm',dpi = 200)

# ------------ prep data for extracting partial effects and relative importance ----------------------
pred_data =
bind_rows(prep_train_baleen%>%dplyr::select(type,forms,balancing,balancing_dat,model_out)%>%
                    unnest(model_out)%>%
                    dplyr::select(type,forms,balancing,balancing_dat,opt_tune)
)

pred_data = pred_data%>%
  unnest(opt_tune)%>%
  dplyr::select(-.config)%>%
  mutate(trees = as.list(trees))

# ------- utility function fit random forest ------------
custom_rf_occ <- function(forms,trees,al_data){


  rf_fit <- ranger::ranger(formula = formula(forms),data = al_data,num.trees = trees,
                           importance = 'impurity',probability = TRUE)

  all_model_form=forms%>%str_split(string = .,pattern = '[~|+]')%>%unlist()

  # importance p-values
  imp_pvalues = importance_pvalues(rf_fit, method =  'altmann',formula = formula(forms),data =
al_data)
  # Compute partial dependence plot
  pred_wrapper <- function(object, newdata) {
    p <- predict(object, data = newdata)$predictions[, 2]
    c("avg" = mean(p), "avg-1sd" = mean(p) - sd(p), "avg+1sd" = mean(p) + sd(p))
  }

  tmp_tibble = tibble(preds = all_model_form[-1],partial = vector('list',length=length(all_model_form[-
1])))
  for(i in 1:nrow(tmp_tibble)){
    if(tmp_tibble$preds[i]=='Month'){
      pred_grid = tibble(Month = 1:12)
      tmp_tibble$partial[[i]] = pdp::partial(object = rf_fit,pred.var = tmp_tibble$preds[i], which.class="1",
                                       prob = TRUE,train = al_data,pred.grid = pred_grid)
    }else if(tmp_tibble$preds[i]=='Hour'){
      pred_grid = tibble(Hour = 0:23)
      tmp_tibble$partial[[i]] = pdp::partial(object = rf_fit,pred.var = tmp_tibble$preds[i], which.class="1",
                                       prob = TRUE,train = al_data,pred.grid = pred_grid)
    }else{
      tmp_tibble$partial[[i]] = pdp::partial(object = rf_fit,pred.var = tmp_tibble$preds[i], which.class="1",
                                       prob = TRUE,train = al_data,grid.resolution = 50)
    }

  }
  import_df = imp_pvalues|>
    as.data.frame()|>
    rownames_to_column(var = 'vars')|>
    as_tibble()|>
    mutate(vars = as.character(vars))

  import_df = import_df|>
    mutate(al_partials = list(tmp_tibble))
```

```
  import_df <- import_df%>%mutate(vars = forcats::fct_reorder(.f = vars,.x = importance,.fun =
mean,.desc = TRUE))

  import_df
}


#  ----  random forest based variable importance --------------------------
system.time(rf_based_imp_occ_balanc <- pred_data|>
            #slice(3)|>
            mutate(rf_out =  purrr::pmap(.l = list(forms = forms,tree = trees,al_data = balancing_dat),
                                         .f = custom_rf_occ)))

# ------- process final output -------------------------------------------

load('SPECIFY_FOLDER_SAVE_FILE/final_model_tune_output_occ_balanced_baleen_final.RData')

rf_importance_al_occ <- rf_based_imp_occ_balanc%>%
  dplyr::select(type,balancing,rf_out)%>%
  unnest(cols = rf_out)%>%
  dplyr::select(-al_partials)

rf_importance_al_occ = rf_importance_al_occ%>%
  left_join(species_map, by = c('type' = 'spp_code'))%>%
  group_by(type,balancing,species)%>%
  mutate(importance_scaled = importance/max(importance)*100)%>%
  ungroup()

rf_importance_al_occ_vars=rf_importance_al_occ%>%
                      dplyr::select(vars)|>
                      dplyr::distinct()|>
                      mutate(varsF =c('Month','Hour',"sst",'chl','ssh','windsp','sst','chl','ssh','windsp','sst',
                           'chl','ssh','windsp'))

rf_importance_al_occ_fin=left_join(rf_importance_al_occ,rf_importance_al_occ_vars)

rf_importance_al_occ_fin2<- rf_importance_al_occ_fin%>%
  mutate(varsB = case_when(
    varsF=="chl"~"Chlorophyll-a",
    varsF=="Hour"~"Hour",
    varsF=="Month"~"Month",
    varsF=="sst"~"SST",
    varsF=="ssh"~"SSH",
    varsF=="windsp"~"Wind speed"))

###--- Plot RF model results -----
plot_smote_importance = rf_importance_al_occ_fin2%>%
  dplyr::filter(balancing=='smote')%>%
  mutate(sig_label = case_when(pvalue < 0.05 ~ '*',
                               TRUE ~ 'NS'))|>
  ggplot()+
  geom_bar(aes(x = reorder(varsB, importance_scaled),y = importance_scaled),
           stat = 'identity',position = position_dodge(width = 1), alpha = 0.6)+
  geom_text(mapping = aes(x = reorder(varsB, importance_scaled),
                          y = importance_scaled+2,label=sig_label),
            position = position_dodge(width = 1),size = 7)+
  labs(x =  'Variables',y = 'Index of importance (%)')+
  facet_grid(~species,scales = 'free')+
  theme_bw()+
  coord_flip()+
  theme(strip.background = element_rect(fill = 'white'),
        strip.text = element_text(size = 15.5),
        text = element_text(size=20), axis.title = element_text(size=20),
        axis.text=element_text(size=20),axis.text.x=element_text(size=17),
        panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.spacing = unit(0.3, "cm", data = NULL))
```

```
plot_smote_importance

#--- Save RF model figure -----
ggsave(filename =
'SPECIFY_FOLDER_SAVE_FILE/RF_importance_final_occ_PEIs_baleen_whales_SMOTE.png',
       plot = plot_smote_importance,device = 'png',width = 62,height = 17,dpi = 300,units = 'cm')

# ----- partial dependency plot ------------------------------------------
ren_data = function(dat) {dat%>%as_tibble()%>%dplyr::rename_with(.cols = 1,.fn = function(x) 'vars')}

rf_pdp_models_occ <- rf_based_imp_occ_balanc%>%
  dplyr::select(type,balancing,rf_out)%>%
  unnest(cols = rf_out)%>%
  dplyr::select(-c(vars,importance))%>%
  distinct()%>%
  unnest(cols = al_partials)%>%
  mutate(ren_dat =  purrr::map(.x = partial,.f = ~ren_data(dat = .x)))%>%
  dplyr::select(-partial)%>%
  unnest(cols = ren_dat)

rf_pdp_models_occ = rf_pdp_models_occ%>%
  left_join(species_map, by = c('type' = 'spp_code'))

rf_pdp_models_occ_preds=rf_pdp_models_occ %>%
  dplyr::select(preds)|>
  dplyr::distinct()|>
  mutate(predsF =c('Month','Hour',"sst",'chl','ssh','windsp','sst','chl','ssh','windsp','sst',
                   'chl','ssh','windsp'))

rf_pdp_models_occ_fin=left_join(rf_pdp_models_occ,rf_pdp_models_occ_preds)

rf_pdp_models_occ_fin2<- rf_pdp_models_occ_fin%>%
  mutate(predsB = case_when(
    predsF=="chl"~"Chlorophyll-a",
    predsF=="Hour"~"Hour",
    predsF=="Month"~"Month",
    predsF=="sst"~"SST",
    predsF=="ssh"~"SSH",
    predsF=="windsp"~"Wind speed"))

#---- Plot RF model results -----
partial_smote_plt = rf_pdp_models_occ_fin2%>%
  dplyr::filter(balancing=='smote')%>%
  #dplyr::filter(balancing%in%c('smote','adasyn'))%>%
  ggplot()+
  geom_line(aes(x = vars,y = yhat), size=1.2)+
  scale_linetype_manual(name = 'variable',values = c( "solid","dotted",'dashed'))+
  #guides(color=preds)+
  labs(y = 'Partial effect', x = 'Values of predictor variables')+
  facet_grid(species~predsB, scale = 'free')+
  theme_bw()+
  theme(strip.background = element_rect(fill = 'white'),
        legend.position = c(0.86, 0.1),legend.title=element_blank(),
        strip.text.x = element_text(size = 15.2),
        strip.text.y = element_text(size = 20),
        text = element_text(size=20), axis.title = element_text(size=20),
        axis.text=element_text(size=20),axis.text.x=element_text(size=15),
        panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.spacing = unit(0.3, "cm", data = NULL))

partial_smote_plt

#--- Save RF model -----
ggsave(filename =
'C:/Users/fanni/Documents/MARION_ISLAND/RF_partial_effect_plot_PEIs_baleen_whales_SMOTE.
png',
```

```
      plot = partial_smote_plt,device = 'png',width = 85,height = 70,units = 'cm',dpi = 200)

# ---------   write  tuning and model output ------------------------------
save(list = c('rf_pdp_models_occ','rf_importance_al_occ','rf_based_imp_occ_balanc','all_calibs',
              'all_test_perf_occ'),file =
'SPECIFY_FOLDER_SAVE_FILE/final_model_tune_output_occ_balanced_baleen_final.RData')

#---- Plot ADASYN results ----

partial_adasyn_plt = rf_pdp_models_occ_fin2%>%
  dplyr::filter(balancing=='adasyn')%>%
  #dplyr::filter(balancing%in%c('smote','adasyn'))%>%
  ggplot()+
  geom_line(aes(x = vars,y = yhat), size=1.2)+
  scale_linetype_manual(name = 'variable',values = c( "solid","dotted",'dashed'))+
  #guides(color=preds)+
  labs(y = 'Partial effect', x = 'Values of predictor variables')+
  facet_grid(species~predsB, scale = 'free')+
  theme_bw()+
  theme(strip.background = element_rect(fill = 'white'),
        legend.position = c(0.86, 0.1),legend.title=element_blank(),
        strip.text.x = element_text(size = 15.2),
        strip.text.y = element_text(size = 20),
        text = element_text(size=20), axis.title = element_text(size=20),
        axis.text=element_text(size=20),axis.text.x=element_text(size=15),
        panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.spacing = unit(0.3, "cm", data = NULL))

partial_adasyn_plt

ggsave(filename =
'SPECIFY_FOLDER_SAVE_FILE/RF_partial_effect_plot_PEIs_baleen_whales_ADASYN.png',
       plot = partial_adasyn_plt,device = 'png',width = 85,height = 70,units = 'cm',dpi = 200)

plot_adasyn_importance = rf_importance_al_occ_fin2%>%
  dplyr::filter(balancing=='adasyn')%>%
  mutate(sig_label = case_when(pvalue < 0.05 ~ '*',
                               TRUE ~ 'NS'))|>
  ggplot()+
  geom_bar(aes(x = reorder(varsB, importance_scaled),y = importance_scaled),
           stat = 'identity',position = position_dodge(width = 1), alpha = 0.6)+
  geom_text(mapping = aes(x = reorder(varsB, importance_scaled),
                          y = importance_scaled+2,label=sig_label),
            position = position_dodge(width = 1),size = 7)+
  labs(x =  'Variables',y = 'Index of importance (%)')+
  facet_grid(~species,scales = 'free')+
  theme_bw()+
  coord_flip()+
  theme(strip.background = element_rect(fill = 'white'),
        #legend.position = c(0.46, 0.71),legend.title=element_blank(),
        strip.text = element_text(size = 15.5),
        text = element_text(size=20), axis.title = element_text(size=20),
        axis.text=element_text(size=20),axis.text.x=element_text(size=17),
        panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.spacing = unit(0.3, "cm", data = NULL))

plot_adasyn_importance

ggsave(filename =
'SPECIFY_FOLDER_SAVE_FILE/RF_importance_final_occ_PEIs_baleen_whales_ADASYN.png',
       plot = plot_adasyn_importance,device = 'png',width = 62,height = 17,dpi = 300,units = 'cm')
```