```
# --------  compute and get vals for calibration plot ---------------------
calib_plot_vals = function(the_data,species_name){
  calib_dat = the_data%>%
    filter(species%in%species_name)%>%
    dplyr::select(-species)%>%
    mutate(bin = ntile(.pred_1, 10)) %>%
    # Bin prediction into 10ths
    group_by(bin) %>%
    mutate(n = n(), # Get ests and CIs
           bin_pred = mean(.pred_1),
           bin_prob = mean(as.numeric(as.character(values))),
           se = sqrt((bin_prob * (1 - bin_prob)) / n),
           ul = bin_prob + 1.96 * se,
           ll = bin_prob - 1.96 * se)%>%ungroup()
calib_dat
}


# ---------  plot  calibration -------------------------------------------
gg_calib_plot = function(dat,main_title){
  dat%>%
  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, 1, by = 0.1)) +
    scale_x_continuous(limits = c(0, 1), breaks = seq(0, 1, by = 0.1)) +
    geom_abline() + # 45 degree line indicating perfect calibration
    geom_smooth(data = dat,method = "lm", se = FALSE, linetype = "dashed",
              color = "black", formula = y~-1 + x) +
    # straight line fit through estimates
    geom_smooth(data = dat,aes(x = .pred_1, y = as.numeric(as.character(values))),
              color = "red", se = FALSE, method = "loess") +
    # loess fit through estimates
    labs(x = '',ylab = "Observed Probability",subtitle = main_title)+
    theme_bw()

}

# -----------    random forest for tuning and testing clasificaiton -------
cust_rf_tune_test_class <- function(the_recipe,train_rsample,final_split){

  # ------ set up the classification model ---------------------------------
  the_models <- rand_forest()%>%set_args(trees = tune())%>%set_engine('ranger')%>%
    set_mode('classification')

  # ------- setup tuning grid ---------------------------------------------
  tune_grd <- expand.grid(trees = c(100,200,500,1000,1500,2000))

  #  prepare the model workflow
  models_workflow <- workflow()%>%add_recipe(the_recipe)%>%add_model(the_models)

  #  -------- start tuning -------------------------------------------------
  cat('start tuning random forest..... \n')

  models_tune <- models_workflow%>%
    tune_grid(grid = tune_grd,resamples = train_rsample,
            metrics = metric_set(yardstick::accuracy,
yardstick::sensitivity,yardstick::specificity,yardstick::roc_auc),
            control =  control_grid(pkgs = c('ranger','yardstick')))

  cat('completed tuning random forest..... \n')
  # -----  extract optimal tuning parameter --------------------------------
  models_final <- models_tune%>%
    select_best(metric = "roc_auc")

  # --------- finalize workflow -------------------------------------------
  workflow_final <- models_workflow%>%
    finalize_workflow(models_final)

  # ------- fit the model on all the training set and evaluate on test set ----------------------
  fit_final <- workflow_final %>%
    last_fit(final_split)
  # ------ evalutate model on test set -------------------------------------
  test_perf <- fit_final %>% collect_metrics()%>%mutate(model = 'random forest')

  # generate predictions from the test set
  test_preds_dat <- fit_final%>%dplyr::select(.predictions)%>%unnest(.predictions)

  rslt <- tibble(opt_tune = list(models_final),all_tune = list(models_tune),
                 test_perf = list(test_perf),test_pred = list(test_preds_dat))
  rslt

}
```