```
## Supplemental R file for manuscript ID TiH-S2024-0074 entitled
  # "Explore a cost-effective way for nutrient management with
  # machine learning for container plants"
## Authors: Ping Yu & Kuan Qin

######################################################################
######################################################################
# Machine learning part
library(tidyverse)
library(tidymodels)
library(skimr)
library(tibble)
library(knitr)
library(workflows)
library(randomForest)
library(kknn)
library(naivebayes)
library(e1071)
library("writexl")

######################################################################
########## Read data
dta <- read.csv("Basil-reform.csv")
dta$Nlevel <- as.factor(dta$Nlevel)

DT <- data.frame()
RF <- data.frame()
KNN <- data.frame()
NB <- data.frame()
SVM <- data.frame()

for (i in 1:(nrow(dta)/25)) {
  order1 <- (i-1)*25+1
  order2 <- i*25
  x <- dta[order1:order2,c(2:5,13)]
  # Create a train/test data split
  set.seed(seed = 123)
  game_split <- rsample::initial_split(data = x, prop = 0.70,strata = Nlevel)
  game_train <- training(game_split)
  game_test <- testing(game_split)
  # Prep the recipe on the training data
  recipe_prepped <- recipe(Nlevel ~ ., data = game_train) %>%
    prep()

  # Bake the recipe (i.e. apply the preprocessing to the data)
  train_baked <- bake(recipe_prepped, new_data = game_train)
  test_baked  <- bake(recipe_prepped, new_data = game_test)

  # DECISION TREE
  # Fit the decision tree model
  game_dt <-
    decision_tree(mode = "classification") %>%
    set_engine("rpart") %>%
    fit(Nlevel ~ ., data = game_train)
  # Model assessment
  ## Create a vector of predictions and true target values
  predictions_dt <- game_dt %>%
    predict(new_data = game_test) %>%
    bind_cols(test_baked %>% select(Nlevel))
  # Calculate a confusion matrix
  predictions_dt %>%
    conf_mat(Nlevel, .pred_class) %>%
    pluck(1) %>%
    as_tibble() %>%
```

```
    ggplot(aes(Prediction, Truth, alpha = n)) +
    geom_tile(show.legend = FALSE) +
    geom_text(aes(label = n), colour = "white", alpha = 1, size = 8)
  # Determine the accuracy of the model
  DT[i,1:2] <- predictions_dt %>%
    metrics(Nlevel, .pred_class) %>%
    select(-.estimator) %>%
    filter(.metric == "accuracy")

  # Next try Random forest.
  game_model <- randomForest(Nlevel ~ ., data = game_train)
  # Creating the confusion matrix
  prediction <- predict(game_model, newdata = game_test)
  a <- table(prediction, game_test$Nlevel)
  RF[i,1] <- c(sum(diag(a))/sum(a))

  # Next I'll try KNN.

  # Define the recipe
  recipe_knn <-
    recipe(Nlevel ~ ., data = x) %>%
    step_normalize(all_numeric()) %>%
    step_impute_knn(all_predictors())

  # Specify the Model
  # For the k-nearest neighbor
  knn_model <-
    nearest_neighbor() %>%
    set_args(ks = 4) %>%
    set_engine("kknn") %>%
    set_mode("classification")

  # set the workflow
  knn_workflow <- workflow() %>%
    # add the recipe
    add_recipe(recipe_knn) %>%
    # add the model
    add_model(knn_model)

  # Evaluate the model on the test set
  knn_fit <- knn_workflow %>%
    # fit on the training set and evaluate on test set
    last_fit(game_split)

  KNN[i,1:2] <- knn_fit %>%
    collect_metrics() %>%
    select(-.estimator,-.config) %>%
    filter(.metric == "accuracy")

  # Next, I'll try Naive Bayes model

  nbClassifier=naive_bayes(Nlevel ~ . , usekernel=T, data=game_train)
  # Creating the confusion matrix
  prediction <- predict(nbClassifier, newdata = game_test)
  a<-table(prediction, game_test$Nlevel)
  NB[i,1] <- c(sum(diag(a))/sum(a))

  # Next, I'll try SVM.

  svmfit = svm(Nlevel ~ ., data = game_train, kernel = "linear", type = 'C-
classification')
  # Predict
  y_pred = predict(svmfit, newdata = game_test)
  a<-table(y_pred, game_test$Nlevel)
  SVM[i,1] <- c(sum(diag(a))/sum(a))
```

```
}

#############################################
combine <- cbind(data.frame(DT[,2]), RF, data.frame(KNN[,2]), NB, SVM)
write_xlsx(combine,'Basil-accuracy-reform.xlsx')


#########################################################################
########## Read data
dta <- read.csv("Marigold-reform.csv")
dta$Nlevel <- as.factor(dta$Nlevel)

DT <- data.frame()
RF <- data.frame()
KNN <- data.frame()
NB <- data.frame()
SVM <- data.frame()

for (i in 1:(nrow(dta)/25)) {
  order1 <- (i-1)*25+1
  order2 <- i*25
  x <- dta[order1:order2,c(2:5,13)]
  # Create a train/test data split
  set.seed(seed = 123)
  game_split <- rsample::initial_split(data = x, prop = 0.70,strata = Nlevel)
  game_train <- training(game_split)
  game_test <- testing(game_split)
  # Prep the recipe on the training data
  recipe_prepped <- recipe(Nlevel ~ ., data = game_train) %>%
    prep()

  # Bake the recipe (i.e. apply the preprocessing to the data)
  train_baked <- bake(recipe_prepped, new_data = game_train)
  test_baked  <- bake(recipe_prepped, new_data = game_test)

  # DECISION TREE
  # Fit the decision tree model
  game_dt <-
    decision_tree(mode = "classification") %>%
    set_engine("rpart") %>%
    fit(Nlevel ~ ., data = game_train)
  # Model assessment
  ## Create a vector of predictions and true target values
  predictions_dt <- game_dt %>%
    predict(new_data = game_test) %>%
    bind_cols(test_baked %>% select(Nlevel))
  # Calculate a confusion matrix
  predictions_dt %>%
    conf_mat(Nlevel, .pred_class) %>%
    pluck(1) %>%
    as_tibble() %>%
    ggplot(aes(Prediction, Truth, alpha = n)) +
    geom_tile(show.legend = FALSE) +
    geom_text(aes(label = n), colour = "white", alpha = 1, size = 8)
  # Determine the accuracy of the model
  DT[i,1:2] <- predictions_dt %>%
    metrics(Nlevel, .pred_class) %>%
    select(-.estimator) %>%
    filter(.metric == "accuracy")

  # Next try Random forest.
  game_model <- randomForest(Nlevel ~ ., data = game_train)
  # Creating the confusion matrix
  prediction <- predict(game_model, newdata = game_test)
  a <- table(prediction, game_test$Nlevel)
```

```
  RF[i,1] <- c(sum(diag(a))/sum(a))

  # Next I'll try KNN.

  # Define the recipe
  recipe_knn <-
    recipe(Nlevel ~ ., data = x) %>%
    step_normalize(all_numeric()) %>%
    step_impute_knn(all_predictors())

  # Specify the Model
  # For the k-nearest neighbor
  knn_model <-
    nearest_neighbor() %>%
    set_args(ks = 4) %>%
    set_engine("kknn") %>%
    set_mode("classification")

  # set the workflow
  knn_workflow <- workflow() %>%
    # add the recipe
    add_recipe(recipe_knn) %>%
    # add the model
    add_model(knn_model)

  # Evaluate the model on the test set
  knn_fit <- knn_workflow %>%
    # fit on the training set and evaluate on test set
    last_fit(game_split)

  KNN[i,1:2] <- knn_fit %>%
    collect_metrics() %>%
    select(-.estimator,-.config) %>%
    filter(.metric == "accuracy")

  # Next, I'll try Naive Bayes model

  nbClassifier=naive_bayes(Nlevel ~ . , usekernel=T, data=game_train)
  # Creating the confusion matrix
  prediction <- predict(nbClassifier, newdata = game_test)
  a<-table(prediction, game_test$Nlevel)
  NB[i,1] <- c(sum(diag(a))/sum(a))

  # Next, I'll try SVM.

  svmfit = svm(Nlevel ~ ., data = game_train, kernel = "linear", type = 'C-
classification')
  # Predict
  y_pred = predict(svmfit, newdata = game_test)
  a<-table(y_pred, game_test$Nlevel)
  SVM[i,1] <- c(sum(diag(a))/sum(a))
}

################################################
combine <- cbind(data.frame(DT[,2]), RF, data.frame(KNN[,2]), NB, SVM)
write_xlsx(combine,'Marigold-accuracy-reform.xlsx')


#####################################################################
########## Read data
dta <- read.csv("Pepper-reform.csv")
dta$Nlevel <- as.factor(dta$Nlevel)

DT <- data.frame()
RF <- data.frame()
```

```
KNN <- data.frame()
NB <- data.frame()
SVM <- data.frame()

for (i in 1:(nrow(dta)/25)) {
  order1 <- (i-1)*25+1
  order2 <- i*25
  x <- dta[order1:order2,c(2:5,13)]
  # Create a train/test data split
  set.seed(seed = 123)
  game_split <- rsample::initial_split(data = x, prop = 0.70,strata = Nlevel)
  game_train <- training(game_split)
  game_test <- testing(game_split)
  # Prep the recipe on the training data
  recipe_prepped <- recipe(Nlevel ~ ., data = game_train) %>%
    prep()

  # Bake the recipe (i.e. apply the preprocessing to the data)
  train_baked <- bake(recipe_prepped, new_data = game_train)
  test_baked  <- bake(recipe_prepped, new_data = game_test)

  # DECISION TREE
  # Fit the decision tree model
  game_dt <-
    decision_tree(mode = "classification") %>%
    set_engine("rpart") %>%
    fit(Nlevel ~ ., data = game_train)
  # Model assessment
  ## Create a vector of predictions and true target values
  predictions_dt <- game_dt %>%
    predict(new_data = game_test) %>%
    bind_cols(test_baked %>% select(Nlevel))
  # Calculate a confusion matrix
  predictions_dt %>%
    conf_mat(Nlevel, .pred_class) %>%
    pluck(1) %>%
    as_tibble() %>%
    ggplot(aes(Prediction, Truth, alpha = n)) +
    geom_tile(show.legend = FALSE) +
    geom_text(aes(label = n), colour = "white", alpha = 1, size = 8)
  # Determine the accuracy of the model
  DT[i,1:2] <- predictions_dt %>%
    metrics(Nlevel, .pred_class) %>%
    select(-.estimator) %>%
    filter(.metric == "accuracy")

  # Next try Random forest.
  game_model <- randomForest(Nlevel ~ ., data = game_train)
  # Creating the confusion matrix
  prediction <- predict(game_model, newdata = game_test)
  a <- table(prediction, game_test$Nlevel)
  RF[i,1] <- c(sum(diag(a))/sum(a))

  # Next I'll try KNN.

  # Define the recipe
  recipe_knn <-
    recipe(Nlevel ~ ., data = x) %>%
    step_normalize(all_numeric()) %>%
    step_impute_knn(all_predictors())

  # Specify the Model
  # For the k-nearest neighbor
  knn_model <-
    nearest_neighbor() %>%
```

```
    set_args(ks = 4) %>%
    set_engine("kknn") %>%
    set_mode("classification")

  # set the workflow
  knn_workflow <- workflow() %>%
    # add the recipe
    add_recipe(recipe_knn) %>%
    # add the model
    add_model(knn_model)

  # Evaluate the model on the test set
  knn_fit <- knn_workflow %>%
    # fit on the training set and evaluate on test set
    last_fit(game_split)

  KNN[i,1:2] <- knn_fit %>%
    collect_metrics() %>%
    select(-.estimator,-.config) %>%
    filter(.metric == "accuracy")

  # Next, I'll try Naive Bayes model

  nbClassifier=naive_bayes(Nlevel ~ . , usekernel=T, data=game_train)
  # Creating the confusion matrix
  prediction <- predict(nbClassifier, newdata = game_test)
  a<-table(prediction, game_test$Nlevel)
  NB[i,1] <- c(sum(diag(a))/sum(a))

  # Next, I'll try SVM.

  svmfit = svm(Nlevel ~ ., data = game_train, kernel = "linear", type = 'C-
classification')
  # Predict
  y_pred = predict(svmfit, newdata = game_test)
  a<-table(y_pred, game_test$Nlevel)
  SVM[i,1] <- c(sum(diag(a))/sum(a))
}

################################################
combine <- cbind(data.frame(DT[,2]), RF, data.frame(KNN[,2]), NB, SVM)
write_xlsx(combine,'Pepper-accuracy-reform.xlsx')


#####################################################################
########## Read data
dta <- read.csv("Sage-reform.csv")
dta$Nlevel <- as.factor(dta$Nlevel)

DT <- data.frame()
RF <- data.frame()
KNN <- data.frame()
NB <- data.frame()
SVM <- data.frame()

for (i in 1:(nrow(dta)/25)) {
  order1 <- (i-1)*25+1
  order2 <- i*25
  x <- dta[order1:order2,c(2:5,13)]
  # Create a train/test data split
  set.seed(seed = 123)
  game_split <- rsample::initial_split(data = x, prop = 0.70,strata = Nlevel)
  game_train <- training(game_split)
  game_test <- testing(game_split)
  # Prep the recipe on the training data
```

```
recipe_prepped <- recipe(Nlevel ~ ., data = game_train) %>%
  prep()

# Bake the recipe (i.e. apply the preprocessing to the data)
train_baked <- bake(recipe_prepped, new_data = game_train)
test_baked  <- bake(recipe_prepped, new_data = game_test)

# DECISION TREE
# Fit the decision tree model
game_dt <-
  decision_tree(mode = "classification") %>%
  set_engine("rpart") %>%
  fit(Nlevel ~ ., data = game_train)
# Model assessment
## Create a vector of predictions and true target values
predictions_dt <- game_dt %>%
  predict(new_data = game_test) %>%
  bind_cols(test_baked %>% select(Nlevel))
# Calculate a confusion matrix
predictions_dt %>%
  conf_mat(Nlevel, .pred_class) %>%
  pluck(1) %>%
  as_tibble() %>%
  ggplot(aes(Prediction, Truth, alpha = n)) +
  geom_tile(show.legend = FALSE) +
  geom_text(aes(label = n), colour = "white", alpha = 1, size = 8)
# Determine the accuracy of the model
DT[i,1:2] <- predictions_dt %>%
  metrics(Nlevel, .pred_class) %>%
  select(-.estimator) %>%
  filter(.metric == "accuracy")

# Next try Random forest.
game_model <- randomForest(Nlevel ~ ., data = game_train)
# Creating the confusion matrix
prediction <- predict(game_model, newdata = game_test)
a <- table(prediction, game_test$Nlevel)
RF[i,1] <- c(sum(diag(a))/sum(a))

# Next I'll try KNN.

# Define the recipe
recipe_knn <-
  recipe(Nlevel ~ ., data = x) %>%
  step_normalize(all_numeric()) %>%
  step_impute_knn(all_predictors())

# Specify the Model
# For the k-nearest neighbor
knn_model <-
  nearest_neighbor() %>%
  set_args(ks = 4) %>%
  set_engine("kknn") %>%
  set_mode("classification")

# set the workflow
knn_workflow <- workflow() %>%
  # add the recipe
  add_recipe(recipe_knn) %>%
  # add the model
  add_model(knn_model)

# Evaluate the model on the test set
knn_fit <- knn_workflow %>%
  # fit on the training set and evaluate on test set
```

```
    last_fit(game_split)

  KNN[i,1:2] <- knn_fit %>%
    collect_metrics() %>%
    select(-.estimator,-.config) %>%
    filter(.metric == "accuracy")

  # Next, I'll try Naive Bayes model

  nbClassifier=naive_bayes(Nlevel ~ . , usekernel=T, data=game_train)
  # Creating the confusion matrix
  prediction <- predict(nbClassifier, newdata = game_test)
  a<-table(prediction, game_test$Nlevel)
  NB[i,1] <- c(sum(diag(a))/sum(a))

  # Next, I'll try SVM.

  svmfit = svm(Nlevel ~ ., data = game_train, kernel = "linear", type = 'C-
classification')
  # Predict
  y_pred = predict(svmfit, newdata = game_test)
  a<-table(y_pred, game_test$Nlevel)
  SVM[i,1] <- c(sum(diag(a))/sum(a))
}

################################################
combine <- cbind(data.frame(DT[,2]), RF, data.frame(KNN[,2]), NB, SVM)
write_xlsx(combine,'Sage-accuracy-reform.xlsx')
```