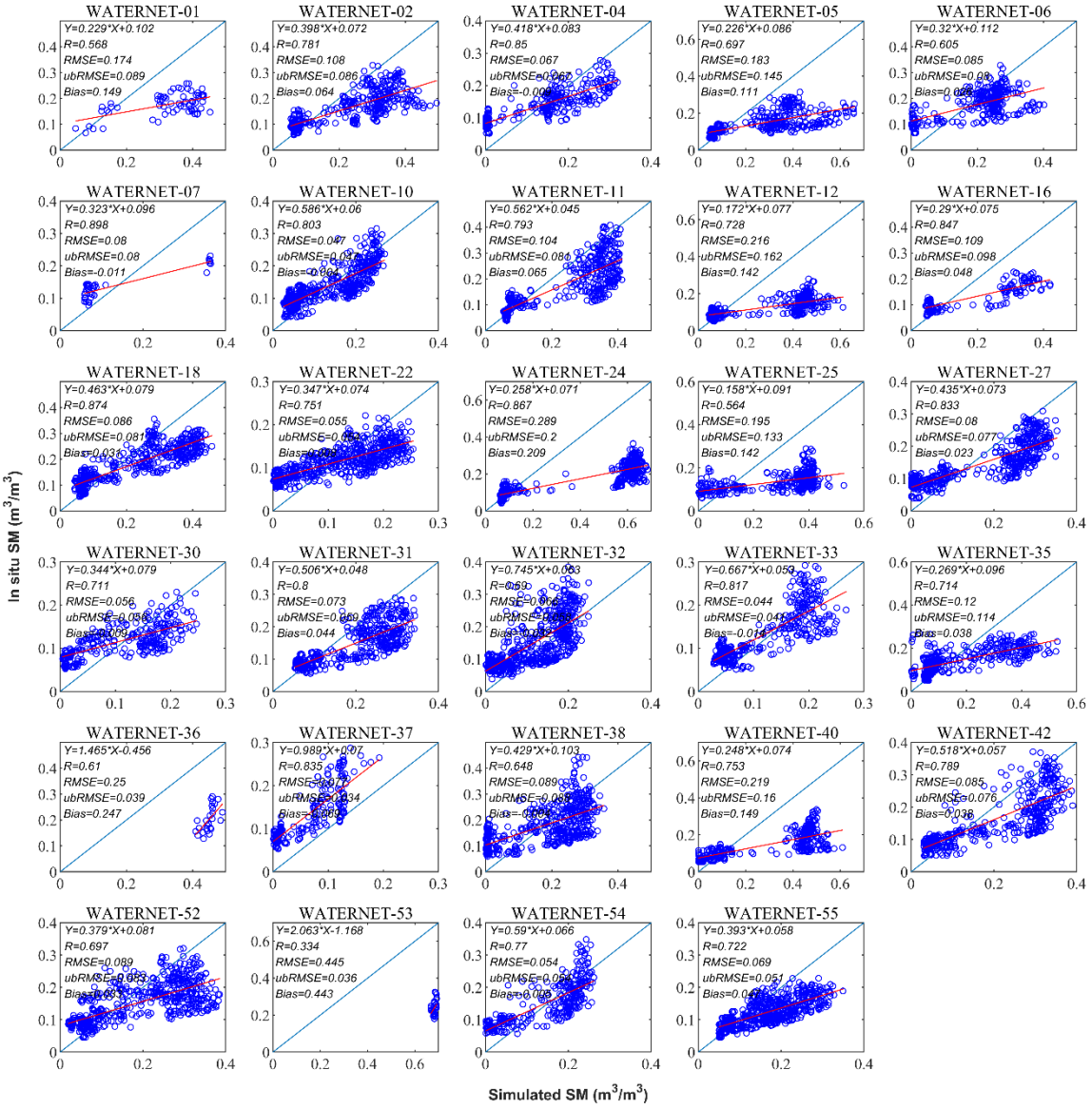

Figure S1: Comparison between the downscaled SM and in situ SM of Babao Network.

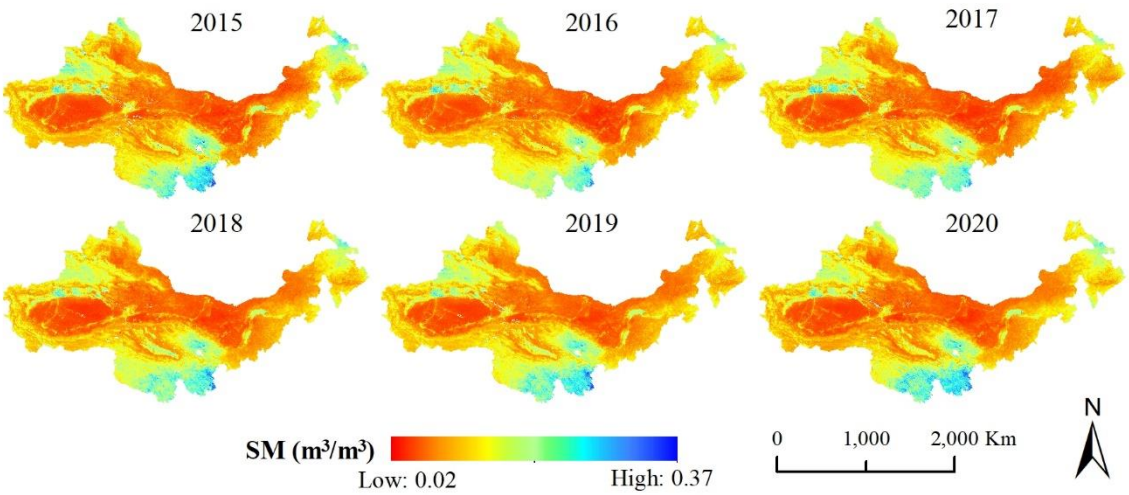

Figure S2: Annual average SM in the study area. Note: The SM data for 2015 are only from 2015/3/22.

CODES: The codes were written using RStudio based on the R language.

(1) The codes for sample selection, PATH present the path of the file.

```
##Selection of valid samples for the study area
rm(list = ls())
library(xlsx)
library(randomForest)
library(varSelRF)
library(pROC)
library(raster)
library(rgdal)
library(fpp2)
library(tcltk)
library(forecast)
library(ggplot2)
library(tweedie)
library(e1071)
library(RSNNS)
library(rpart)
# input terrain parameters
DEMFILE <-
  raster('PATH/DEM/DEM36000.tif')
SLOPEFILE <-
  raster('PATH/DEM/Slope36000.tif')
ASPECTFILE <-
  raster('PATH/DEM/Aspect36000.tif')
#input longitude and latitude，The results show that latitude and longitude have little effect
LONGFILE <-
  raster('PATH/Location/long36000.tif')
```

```
LATFILE <-
   raster('PATH/Location/lat36000.tif')
# input soil parameters
SANDFILE <-
   raster('PATH/soil/SA36000.tif')
SILTFILE <-
   raster('PATH/soil/SI36000.tif')
CLAYFILE <-
   raster('PATH/soil/CL36000.tif')

path <- 'PATH/MODIS/MOD13A2/tif36000/'
docs <- dir(path, '*EVI.tif')
n <- length(docs)
num <- matrix(nrow = n, ncol = 1)
Factors <-
   c(
      'LST',
      'EVI',
      'NDVI',
      'DEM',
      'SLOPE',
      'ASPECT',
      'SAND',
      'SILT',
      'CLAY',
      'LONG',
      'LAT',
      'LAI',
      'FAPAR',
      'NDWI',
      'LSWI',
      'NSDSI',
      'ALBEDO',
      'SM'
   )

for (k in 1:n) {
   iris <- matrix(nrow = 1, ncol = 18)
   iris2 <- matrix(nrow = 1, ncol = 18)
   colnames(iris) <- Factors
   colnames(iris2) <- Factors
   print(docs[k])
```

```
date <- substr(docs[k], 1, 7)
date <- as.numeric(date)    #numeric

# var explained
# read MOD13A2
EVIFILE <- raster(paste0(path, docs[k]))
NDVIFILE <- raster(paste0(path, date, '.1_km_16_days_NDVI.tif'))

#Samples for 16 days
for (i in 1:16) {
   date1 <- date + i - 1
   date2 <- as.Date(as.character(date1), '%Y%j')
   date2 <- gsub("-", "", date2)
   filename1 <-
      paste0('PATH/MODIS/MOD11A1/tif36000/',
             date1,
             '.LST_Day_1km.tif')
   filename2 <-
      paste0('PATH/SMAP/xibei36000/',
             'SMAP_L3_SM_P_',
             date2,
             '_R17000_001.h5.tif')
   filename22 <-
      paste0('PATH/SMAP/xibei36000/',
             'SMAP_L3_SM_P_',
             date2,
             '_R17000_002.h5.tif')
   filename3 <-
      paste0('PATH/MODIS/MCD43D58/tif36000/',
             'MCD43D58.A',
             date1,
             '.tif')

   if (i < 9) {
      LAIFILE <-
         raster(paste0('PATH/MODIS/MOD15A2/tif36000/',
                       date,
                       '.Lai_500m.tif'))
      FAPARFILE <-
         raster(paste0('PATH/MODIS/MOD15A2/tif36000/',
                       date,
                       '.Fpar_500m.tif'))
```

```
  NDWIFILE <-
    raster(paste0('PATH/MODIS/MOD09A1/tif36000/',
                  date,
                  '_NDWI.tif'))
  LSWIFILE <-
    raster(paste0('PATH/MODIS/MOD09A1/tif36000/',
                  date,
                  '_LSWI.tif'))
  NSDSIFILE <-
    raster(paste0('PATH/MODIS/MOD09A1/tif36000/',
                  date,
                  '_NSDSI.tif'))
} else{
  LAIFILE <-
    raster(paste0(
      'PATH/MODIS/MOD15A2/tif36000/',
      date + 8,
      '.Lai_500m.tif'
    ))
  FAPARFILE <-
    raster(paste0(
      'PATH/MODIS/MOD15A2/tif36000/',
      date + 8,
      '.Fpar_500m.tif'
    ))
  NDWIFILE <-
    raster(paste0('PATH/MODIS/MOD09A1/tif36000/',
                  date + 8,
                  '_NDWI.tif'))
  LSWIFILE <-
    raster(paste0('PATH/MODIS/MOD09A1/tif36000/',
                  date + 8,
                  '_LSWI.tif'))
  NSDSIFILE <-
    raster(paste0('PATH/MODIS/MOD09A1/tif36000/',
                  date + 8,
                  '_NSDSI.tif'))
}

if (file.exists(filename2)) {
  SMFILE <- raster(filename2)
} else if (file.exists(filename22))
```

```
  {
    SMFILE <- raster(filename22)
  } else{
    next
  }

  if (file.exists(filename1) & file.exists(filename3)) {
    LSTFILE <- raster(filename1)
    ALBEDO <- raster(filename3)
    LAIFILE[LAIFILE > 100] <- 0
    FAPARFILE[FAPARFILE > 100] <- 0

    ## Selection of valid samples
    iriss <-
      cbind(
        matrix(LSTFILE),
        matrix(EVIFILE),
        matrix(NDVIFILE),
        matrix(DEMFILE),
        matrix(SLOPEFILE),
        matrix(ASPECTFILE),
        matrix(SANDFILE),
        matrix(SILTFILE),
        matrix(CLAYFILE),
        matrix(LONGFILE),
        matrix(LATFILE),
        matrix(LAIFILE),
        matrix(FAPARFILE),
        matrix(NDWIFILE),
        matrix(LSWIFILE),
        matrix(NSDSIFILE),
        matrix(ALBEDO),
        matrix(SMFILE)
      )
    xuhao1 <-
      which(iriss[, 1] > 0 &
              iriss[, 9] > 0 &
              iriss[, 18] >= 0 &
              rowSums(is.na(iriss)) == 0)
    iris <- rbind(iris, iriss[xuhao1,])
  }
}
```

```
  num[k, 1] = length(iris[, 1]) - 1
  # Write valid sample in 'PATH/smaples/'
  write.csv(iris[-1,],
            file = paste0('PATH/smaples/',
                          date,
                          '.csv'))
}
```

(2) The codes for the optimal regression model building and the result prediction, PATH present the path of the file.

```
##1.Regression for daily SM data at 1 km resolution
rm(list = ls())
library(xlsx)
library(randomForest)
library(varSelRF)
library(pROC)
library(raster)
library(rgdal)
library(fpp2)
library(tcltk)
library(forecast)
library(ggplot2)
library(tweedie)
library(e1071)
library(RSNNS)
library(rpart)
# library(keras)
library(caret)
library(xgboost)
pb <- tkProgressBar("Progress", "Doing %", 0, 100)

##Prediction
#Input parameters
DEMFILE2 <-
  raster('PATH/DEM/DEM1000.tif')
```

```
SLOPEFILE2 <-
   raster('PATH/DEM/Slope1000.tif')
ASPECTFILE2 <-
   raster('PATH/DEM/Aspect1000.tif')
#input longitude and latitude
LONGFILE2 <-
   raster('PATH/Location/long1000.tif')
LATFILE2 <-
   raster('PATH/Location/lat1000.tif')
# input soil parameters
SANDFILE2 <-
   raster('PATH/soil/SA1000.tif')
SILTFILE2 <-
   raster('PATH/soil/SI1000.tif')
CLAYFILE2 <-
   raster('PATH/soil/CL1000.tif')


#Path of the samples
path1 <- 'PATH/smaples/ '
docs <- dir(path1, '*.csv')
n <- length(docs)
Factors <-
   c(
      'LST',
      'EVI',
      'NDVI',
      'DEM',
      'SLOPE',
      'ASPECT',
      'SAND',
      'SILT',
      'CLAY',
      'LAI',
      'NDWI',
      'LSWI',
      'NSDSI',
      'ALBEDO'
   )
AC <- matrix(0, nrow = n, ncol = 20)      #Accuracy
Imorptance1 <-
   matrix(0, nrow = 14, ncol = n)      #RF variable importance
```

```
Imorptance2 <-
  matrix(0, nrow = 14, ncol = n)      #xgb variable importance

for (k in 1:n) {
  date <- substr(docs[k], 1, 7)
  date <- as.numeric(date)    #numeric format
  iris <-
    read.csv(paste0(path1, date, '.csv'))
  if (nrow(iris) > 60) {
    ### Divide training samples and test samples
    xuhao1 <-
      createDataPartition(y = iris$SM, p = 0.5, list = FALSE)
    #Training set
    ir <- iris[xuhao1, ]
    #Test set
    ir2 <- iris[-xuhao1, ]

    #Random Forest (RF)
    set.seed(1000)
    gx.rf <-
      randomForest(
        SM ~ LST + EVI + NDVI + DEM + SLOPE + ASPECT + SAND + SILT + CLAY + LAI + NDWI +
LSWI + NSDSI + ALBEDO,
        data = ir,
        important = TRUE,
        ntree = 1000,
        proximity = TRUE,
        mtry = 10
      )
    # print(gx.rf)

    # Training accuracy
    pre_rf <- predict(gx.rf, newdata = ir)
    frecoff <-
      cor.test(pre_rf, ir[, 19], method = "pearson", na.action = na.omit)
    AC[k, 1] <- frecoff$estimate    #The correlation coefficient
    AC[k, 2] <- accuracy(pre_rf, ir[, 19])[1, 2]    #RMSE
    ##validation accuracy
    pre_rf2 <- predict(gx.rf, newdata = ir2)
    frecoff2 <-
      cor.test(pre_rf2, ir2[, 19], method = "pearson", na.action = na.omit)
    AC[k, 3] <- frecoff2$estimate    #The correlation coefficient
```

```
    AC[k, 4] <- accuracy(pre_rf2, ir2[, 19])[1, 2]    #RMSE
    Imorptance1[, k] <- gx.rf$importance


    #Multivariable Linear Regression (MLR)
    gx.mlr <-
      lm(
        SM ~ LST + EVI + NDVI + DEM + SLOPE + ASPECT + SAND + SILT + CLAY + LAI + NDWI +
LSWI + NSDSI + ALBEDO,
        data = ir
      )
    # training accuracy
    pre_lm <- predict(gx.mlr, newdata = ir)
    frecoff <-
      cor.test(pre_lm, ir[, 19], method = "pearson", na.action = na.omit)
    AC[k, 5] <- frecoff$estimate    #The correlation coefficient
    AC[k, 6] <- accuracy(pre_lm, ir[, 19])[1, 2]    #RMSE
    # validation accuracy
    pre_lm2 <- predict(gx.mlr, newdata = ir2)
    frecoff2 <-
      cor.test(pre_lm2, ir2[, 19], method = "pearson", na.action = na.omit)
    AC[k, 7] <- frecoff2$estimate    #The correlation coefficient
    AC[k, 8] <- accuracy(pre_lm2, ir2[, 19])[1, 2]    #RMSE


    ### Artificial neural network (ANN)
    ir_mlp <- as.matrix(ir[, c(-1,-11,-12,-14)])
    df <-
      splitForTrainingAndTest(ir_mlp[, -15], ir_mlp[, 15], ratio = 0)
    # Data standardization
    df <- normTrainingAndTestSet(df)
    gx.mlp    <-
      mlp(
        df$inputsTrain,
        df$targetsTrain,
        size = 15,
        learnFunc = "SCG",
        learnFuncParams = c(0, 0, 0, 0),
        # learnFunc = "Std_Backpropagation", learnFuncParams = c(0.2, 0),
        # learnFunc = "BackpropBatch", learnFuncParams = c(10, 0.1),
        # learnFunc = "Quickprop", learnFuncParams = c(0.1, 2.0, 0.0001, 0.1),
        maxit = 1000,
```

```
      metric = "RMSE",
    )
#summary(gx.mlp)
# training accuracy
pre_mlp <- predict(gx.mlp, newdata = df$inputsTrain)
pre_mlp <- as.numeric(pre_mlp)
frecoff <-
    cor.test(
      pre_mlp,
      df$targetsTrain,
      method = "pearson",
      na.action = na.omit,
      conf.level = 0.95
    )
AC[k, 9] <- frecoff$estimate
AC[k, 10] <- accuracy(pre_mlp, df$targetsTrain)[1, 2]

# validation accuracy
ir_mlp2 <- as.matrix(ir2[, c(-1,-11,-12,-14)])
df2 <-
    splitForTrainingAndTest(ir_mlp2[, -15], ir_mlp2[, 15], ratio = 0)
# Data standardization
df2 <- normTrainingAndTestSet(df2)
pre_mlp2 = predict(gx.mlp, newdata = df2$inputsTrain)
pre_mlp2 <- as.numeric(pre_mlp2)
frecoff2 <-
    cor.test(
      pre_mlp2,
      df2$targetsTrain,
      method = "pearson",
      na.action = na.omit,
      conf.level = 0.95
    )
AC[k, 11] <- frecoff2$estimate
AC[k, 12] <- accuracy(pre_mlp2, df2$targetsTrain)[1, 2]


#Xgboost
gx.xgb <- xgboost(
    data = do.call(cbind, ir[, c(-1, -11, -12, -14, -19)]),
    label = ir[, 19],
    booster = "gbtree",
```

```
    objective = "binary:logistic",
    max.depth = 5,
    eta = 0.5,
    #nthread = 2,
    nround = 1000,
    min_child_weight = 1,
    subsample = 1,
    colsample_bytree = 1,
    num_parallel_tree = 1
)

# summary(gx.xgb)
# training accuracy
pre_mlp <-
    predict(gx.xgb, newdata = do.call(cbind, ir[, c(-1, -11, -12, -14, -19)]))
#pre_mlp <- as.numeric(pre_mlp)
frecoff <-
    cor.test(
        pre_mlp,
        ir[, 19],
        method = "pearson",
        na.action = na.omit,
        conf.level = 0.95
    )
AC[k, 13] <- frecoff$estimate
AC[k, 14] <- accuracy(pre_mlp, ir[, 19])[1, 2]
# validation accuracy
pre_mlp2 <-
    predict(gx.xgb, newdata = do.call(cbind, ir2[, c(-1, -11, -12, -14, -19)]))
pre_mlp2 <- as.numeric(pre_mlp2)
frecoff2 <-
    cor.test(
        pre_mlp2,
        ir2[, 19],
        method = "pearson",
        na.action = na.omit,
        conf.level = 0.95
    )
AC[k, 15] <- frecoff2$estimate
AC[k, 16] <- accuracy(pre_mlp2, ir2[, 19])[1, 2]
# Imorptance2[,k]=xgb.importance(model = gx.xgb)$Gain
```

```
    #Support vector regression (SVR)
    #Data normalization
    ir_svm <- ir
    ir_svm2 <- ir2
    ir_svm[, c(-1,-19)] <-
      scale(ir_svm[, c(-1,-19)], center = TRUE, scale = TRUE)
    ir_svm2[, c(-1,-19)] <-
      scale(ir_svm2[, c(-1,-19)], center = TRUE, scale = TRUE)
    gx.svm <-
      svm(
        SM ~ LST + EVI + NDVI + DEM + SLOPE + ASPECT + SAND + SILT + CLAY + LAI + NDWI +
LSWI + NSDSI + ALBEDO,
        data = ir_svm,
        cost = 1,
        gamma = 0.1
      )
    #summary(svm.fit)
    pre_svm = predict(gx.svm, newdata = ir_svm)
    frecoff3 <-
      cor.test(
        pre_svm,
        ir$SM,
        method = "pearson",
        na.action = na.omit,
        conf.level = 0.95
      )
    AC[k, 17] <- frecoff3$estimate
    AC[k, 18] <- accuracy(pre_svm, ir_svm[, 19])[1, 2]
    # validation accuracy
    pre_svm2 <-
      predict(gx.svm, newdata = do.call(cbind, ir_svm2[, c(-1,-11,-12,-14,-19)]))
    pre_mlp2 <- as.numeric(pre_svm2)
    frecoff2 <-
      cor.test(
        pre_svm2,
        ir2$SM,
        method = "pearson",
        na.action = na.omit,
        conf.level = 0.95
      )
    AC[k, 19] <- frecoff2$estimate
```

```
  AC[k, 20] <- accuracy(pre_svm2, ir_svm2[, 19])[1, 2]
}
```

2.The results of prediction

In the prediction part of results, the optimal model was selected and the dependent variable, namely soil moisture, was obtained using the optimal model.

```
## Select the optimal model based on the average RMSE of training set and test set
rf_rmse <- (AC[k, 2] + AC[k, 4]) / 2
mlr_rmse <- (AC[k, 6] + AC[k, 8]) / 2
mlp_rmse <- (AC[k, 10] + AC[k, 12]) / 2
xgb_rmse <- (AC[k, 14] + AC[k, 16]) / 2
svr_rmse <- (AC[k, 18] + AC[k, 20]) / 2
min_rmse <- min(rf_rmse, mlr_rmse, mlp_rmse, xgb_rmse, svr_rmse)


# var explained
EVIFILE2 <-
  raster(paste0(
    'PATH/MODIS/MOD13A2/tif1000/',
    date,
    '.1_km_16_days_EVI.tif'
  ))
NDVIFILE2 <-
  raster(paste0(
    'PATH/MODIS/MOD13A2/tif1000/',
    date,
    '.1_km_16_days_NDVI.tif'
  ))
for (i in 1:16) {
  date1 <- date + i - 1
  filename1 <-
    paste0('PATH/MODIS/MOD11A1/tif1000/',
           date1,
           '.LST_Day_1km.tif')
  filename2 <-
    paste0('PATH/MODIS/MCD43D58/tif1000/MCD43D58.A',
           date1,
           '.tif')
```

```
if (i < 9) {
   date2 <- date
} else{
   date2 <- date + 8
}

LAIFILE2 <-
   raster(paste0('PATH/MODIS/MOD15A2/tif1000/',
                 date2,
                 '.Lai_500m.tif'))
FAPARFILE2 <-
   raster(paste0('PATH/MODIS/MOD15A2/tif1000/',
                 date2,
                 '.Fpar_500m.tif'))
NDWIFILE2 <-
   raster(paste0('PATH/MODIS/MOD09A1/tif1000/',
                 date2,
                 '_NDWI.tif'))
LSWIFILE2 <-
   raster(paste0('PATH/MODIS/MOD09A1/tif1000/',
                 date2,
                 '_LSWI.tif'))
NSDSIFILE2 <-
   raster(paste0('PATH/MODIS/MOD09A1/tif1000/',
                 date2,
                 '_NSDSI.tif'))

LAIFILE2[LAIFILE2 > 100] <- 0
FAPARFILE2[FAPARFILE2 > 100] <- 0

if (file.exists(filename1)) {
   LSTFILE2 <- raster(filename1)
   ALBEDOFILE2 <- raster(filename2)
   # #Output
   result <- LSTFILE2
   result[result >= 0] <- NaN
   train <-
      cbind(
         matrix(LSTFILE2),
         matrix(EVIFILE2),
         matrix(NDVIFILE2),
         matrix(DEMFILE2),
```

```
      matrix(SLOPEFILE2),
      matrix(ASPECTFILE2),
      matrix(SANDFILE2),
      matrix(SILTFILE2),
      matrix(CLAYFILE2),
      matrix(LAIFILE2),
      matrix(NDWIFILE2),
      matrix(LSWIFILE2),
      matrix(NSDSIFILE2),
      matrix(ALBEDOFILE2)
   )
colnames(train) <- Factors
xuhao2 <-
   which(train[, 1] > 0 &
            train[, 6] > 0 &
            train[, 14] > 0 &
            rowSums(is.na(train)) == 0)
train <- train[xuhao2,]
if (min_rmse == rf_rmse) {
   churn.prediction <- predict(gx.rf, newdata = train)
} else if (min_rmse == mlr_rmse) {
   train<-data.frame(train)
   churn.prediction <- predict(gx.mlr, newdata = train)
} else if (min_rmse == mlp_rmse) {
   train <- as.matrix(train)
   df3 <- splitForTrainingAndTest(train, train[, 14], ratio = 0)
   # Data standardization
   df3 <- normTrainingAndTestSet(df3)
   churn.prediction <- predict(gx.mlp, newdata = df3$inputsTrain)
} else if (min_rmse == xgb_rmse) {
   churn.prediction <- predict(gx.xgb, newdata = train)
} else if (min_rmse == xgb_rmse) {
   train <- scale(train, center = TRUE, scale = TRUE)
   churn.prediction <- predict(gx.svm, newdata = train)
}

#Write the results
result[xuhao2] <- churn.prediction
date0 <- as.Date(as.character(date1), '%Y%j')
date0 = gsub("-", "", date0)
rf <-
   writeRaster(
```

```
            result,
            filename = paste0('PATH/SMAP/xibei10003/',
                              date0,
                              '.tif'),
            overwrite = TRUE
          )
        #Timing
        print(date0)
      }
    }
    print(date)
    ## progress
    info <- sprintf("Progress %d%%", round(k * 100 / n))
    setTkProgressBar(pb, k * 100 / n, sprintf("Doing (%s)", info), info)
}
#Accuracy
write.csv(AC, file = "PATH/validation/AC.csv")
# write.csv(Imorptance1, file = "PATH/validation/importance1.csv")
# write.csv(Imorptance2, file = "PATH/validation/importance2.csv")
close(pb)
```