Random Forest:

Breast_Cancer.R

```
param_set = c(1, 2, 4, 6, 8, 12, 16, 20, 22, 24, 26, 32, 40, 48, 64);
for (ratio in c(0.1, 0.2, 0.3, 0.4, 0.5)) {
   D = read.csv("cancer_data_na_removed.csv", header=F,
colClasses=c(rep('integer', 9), 'factor'));
   nclass = 2;
   col = 'V10';
   form = 'V10 ~ .';
   acc_param = rep(0, ncol(D));
   training_indices = sample(nrow(D), floor(nrow(D) * ratio), replace = FALSE,
prob = NULL);
   training_data <- D[training_indices,];
   testing_data <- D[-training_indices,];

   # Bootstrapping
   for (i in 1:30) {
        bootstrap_indices = sample(length(training_indices), replace = TRUE);
        bootstrap_data = training_data[bootstrap_indices,];
        holdout_data = training_data[-bootstrap_indices,];
      for (k in param_set) {
            if (k > ncol(bootstrap_data)) break;
            rf = randomForest(formula = as.formula(form), data = bootstrap_data,
mtry = k, ntree = 1024);
            predictions = predict(rf, holdout_data);
            result = table(predictions, holdout_data[,col]);
            acc_param[k] = acc_param[k] + sum(sum(result * diag(nclass))) /
sum(sum(result));
      }
   }
   rf = randomForest(formula = as.formula(form), data = training_data, mtry =
which.max(acc_param), ntree = 1024);
   predictions = predict(rf, testing_data);
   result = table(predictions, testing_data[,col]);
   accuracy = sum(sum(result * diag(nclass))) / sum(sum(result));
   cat("Bootstrapping: mtry=", which.max(acc_param), "ratio = ", ratio, " accuracy
= ", accuracy, ".\n");
```

```
   #k-fold cross validation, k = 10
   #Randomly shuffle the data
   acc_param = rep(0, ncol(D));
   training_data<-training_data[sample(nrow(training_data)),]

   #Create 10 equally size folds
   folds <- cut(seq(1,nrow(training_data)),breaks=10,labels=FALSE)


   for (i in 1:10) {
        kfold_testing_indices = which(folds==i,arr.ind=TRUE);
        kfold_testing_data = training_data[kfold_testing_indices,];
        kfold_training_data = training_data[-kfold_testing_indices,];
        for (k in param_set) {
                if (k > ncol(kfold_training_data)) break;
                rf = randomForest(formula = as.formula(form), data =
kfold_training_data, mtry = k, ntree = 1024);
                predictions = predict(rf, kfold_testing_data);
                result = table(predictions, kfold_testing_data[,col]);
                acc_param[k] = acc_param[k] + sum(sum(result * diag(nclass))) /
sum(sum(result));
        }
   }

   rf = randomForest(formula = as.formula(form), data = training_data, mtry =
which.max(acc_param), ntree = 1024);
   predictions = predict(rf, testing_data);
   result = table(predictions, testing_data[,col]);
   accuracy = sum(sum(result * diag(nclass))) / sum(sum(result));
   cat("K-fold: mtry=", which.max(acc_param), "ratio = ", ratio, " accuracy = ",
accuracy, ".\n");
}
```

ForestType_Mapping.R

```
param_set = c(1, 2, 4, 6, 8, 12, 16, 20, 22, 24, 26, 32, 40, 48, 64);
for (ratio in c(0.1, 0.2, 0.3, 0.4, 0.5)) {
   D1 = read.csv("ForestTypeMapping_training.csv", header = T);
   D2 = read.csv("ForestTypeMapping_testing.csv", header = T);
   D = rbind(D1, D2);
   nclass = 4;
   col = 'class';
   form = 'class ~ .';
   acc_param = rep(0, ncol(D));
   training_indices = sample(nrow(D), floor(nrow(D) * ratio), replace = FALSE,
prob = NULL);
   training_data <- D[training_indices,];
   testing_data <- D[-training_indices,];

   # Bootstrapping
   for (i in 1:30) {
        bootstrap_indices = sample(length(training_indices), replace = TRUE);
        bootstrap_data = training_data[bootstrap_indices,];
        holdout_data = training_data[-bootstrap_indices,];
      for (k in param_set) {
            if (k > ncol(bootstrap_data)) break;
            rf = randomForest(formula = as.formula(form), data = bootstrap_data,
mtry = k, ntree = 1024);
            predictions = predict(rf, holdout_data);
            result = table(predictions, holdout_data[,col]);
            acc_param[k] = acc_param[k] + sum(sum(result * diag(nclass))) /
sum(sum(result));
      }
   }
   rf = randomForest(formula = as.formula(form), data = training_data, mtry =
which.max(acc_param), ntree = 1024);
   predictions = predict(rf, testing_data);
   result = table(predictions, testing_data[,col]);
   accuracy = sum(sum(result * diag(nclass))) / sum(sum(result));
   cat("Bootstrapping: mtry=", which.max(acc_param), "ratio = ", ratio, " accuracy
= ", accuracy, ".\n");
```

```
    #k-fold cross validation, k = 10
    #Randomly shuffle the data
    acc_param = rep(0, ncol(D));
    training_data<-training_data[sample(nrow(training_data)),]

    #Create 10 equally size folds
    folds <- cut(seq(1,nrow(training_data)),breaks=10,labels=FALSE)


    for (i in 1:10) {
        kfold_testing_indices = which(folds==i,arr.ind=TRUE);
        kfold_testing_data = training_data[kfold_testing_indices,];
        kfold_training_data = training_data[-kfold_testing_indices,];
        for (k in param_set) {
            if (k > ncol(kfold_training_data)) break;
            rf = randomForest(formula = as.formula(form), data =
kfold_training_data, mtry = k, ntree = 1024);
            predictions = predict(rf, kfold_testing_data);
            result = table(predictions, kfold_testing_data[,col]);
            acc_param[k] = acc_param[k] + sum(sum(result * diag(nclass))) /
sum(sum(result));
        }
    }

    rf = randomForest(formula = as.formula(form), data = training_data, mtry =
which.max(acc_param), ntree = 1024);
    predictions = predict(rf, testing_data);
    result = table(predictions, testing_data[,col]);
    accuracy = sum(sum(result * diag(nclass))) / sum(sum(result));
    cat("K-fold: mtry=", which.max(acc_param), "ratio = ", ratio, " accuracy = ",
accuracy, ".\n");
}
```

Optical_Handwritten.R

```
param_set = c(1, 2, 4, 6, 8, 12, 16, 20, 22, 24, 26, 32, 40, 48, 64);
for (ratio in c(0.1, 0.2, 0.3, 0.4, 0.5)) {
   D1 = read.csv("optdigits.tra", header=F, colClasses=c(rep('integer', 64),
'factor'));
   D2 = read.csv("optdigits.tes.txt", header = F);
   D = rbind(D1, D2);
   nclass = 10;
   col = 'V65';
   form = 'V65 ~ .';
   acc_param = rep(0, ncol(D));
   training_indices = sample(nrow(D), floor(nrow(D) * ratio), replace = FALSE,
prob = NULL);
   training_data <- D[training_indices,];
   testing_data <- D[-training_indices,];

   # Bootstrapping
   for (i in 1:30) {
        bootstrap_indices = sample(length(training_indices), replace = TRUE);
        bootstrap_data = training_data[bootstrap_indices,];
        holdout_data = training_data[-bootstrap_indices,];
       for (k in param_set) {
            if (k > ncol(bootstrap_data)) break;
            rf = randomForest(formula = as.formula(form), data = bootstrap_data,
mtry = k, ntree = 1024);
            predictions = predict(rf, holdout_data);
            result = table(predictions, holdout_data[,col]);
            acc_param[k] = acc_param[k] + sum(sum(result * diag(nclass))) /
sum(sum(result));
       }
   }
   rf = randomForest(formula = as.formula(form), data = training_data, mtry =
which.max(acc_param), ntree = 1024);
   predictions = predict(rf, testing_data);
   result = table(predictions, testing_data[,col]);
   accuracy = sum(sum(result * diag(nclass))) / sum(sum(result));
   cat("Bootstrapping: mtry=", which.max(acc_param), "ratio = ", ratio, " accuracy
= ", accuracy, ".\n");
```

```
   #k-fold cross validation, k = 10
   #Randomly shuffle the data
   acc_param = rep(0, ncol(D));
   training_data<-training_data[sample(nrow(training_data)),]

   #Create 10 equally size folds
   folds <- cut(seq(1,nrow(training_data)),breaks=10,labels=FALSE)


   for (i in 1:10) {
        kfold_testing_indices = which(folds==i,arr.ind=TRUE);
        kfold_testing_data = training_data[kfold_testing_indices,];
        kfold_training_data = training_data[-kfold_testing_indices,];
        for (k in param_set) {
                if (k > ncol(kfold_training_data)) break;
                rf = randomForest(formula = as.formula(form), data =
kfold_training_data, mtry = k, ntree = 1024);
                predictions = predict(rf, kfold_testing_data);
                result = table(predictions, kfold_testing_data[,col]);
                acc_param[k] = acc_param[k] + sum(sum(result * diag(nclass))) /
sum(sum(result));
        }
   }

   rf = randomForest(formula = as.formula(form), data = training_data, mtry =
which.max(acc_param), ntree = 1024);
   predictions = predict(rf, testing_data);
   result = table(predictions, testing_data[,col]);
   accuracy = sum(sum(result * diag(nclass))) / sum(sum(result));
   cat("K-fold: mtry=", which.max(acc_param), "ratio = ", ratio, " accuracy = ",
accuracy, ".\n");
}
```

Iris.R

```
param_set = c(1, 2, 4, 6, 8, 12, 16, 20, 22, 24, 26, 32, 40, 48, 64);
for (ratio in c(0.1, 0.2, 0.3, 0.4, 0.5)) {
   D = read.csv("Iris.csv", header=F);
   nclass = 3;
   col = 'V5';
   form = 'V5 ~ .';
   acc_param = rep(0, ncol(D));
   training_indices1 = sample(c(1:50), floor(50 * ratio), replace = FALSE, prob =
NULL);
   training_indices2 = sample(c(51:100), floor(50 * ratio), replace = FALSE, prob
= NULL);
   training_indices3 = sample(c(101:150), floor(50 * ratio), replace = FALSE, prob
= NULL);
   training_indices = c(training_indices1, training_indices2, training_indices3);
   training_data <- D[training_indices,];
   testing_data <- D[-training_indices,];

   # Bootstrapping
   for (i in 1:30) {
        bootstrap_indices = sample(length(training_indices), replace = TRUE);
        bootstrap_data = training_data[bootstrap_indices,];
        holdout_data = training_data[-bootstrap_indices,];
       for (k in param_set) {
            if (k >= ncol(bootstrap_data)) break;
            rf = randomForest(formula = as.formula(form), data = bootstrap_data,
mtry = k, ntree = 1024);
            predictions = predict(rf, holdout_data);
            result = table(predictions, holdout_data[,col]);
            acc_param[k] = acc_param[k] + sum(sum(result * diag(nclass))) /
sum(sum(result));
       }
   }
   rf = randomForest(formula = as.formula(form), data = training_data, mtry =
which.max(acc_param), ntree = 1024);
   predictions = predict(rf, testing_data);
   result = table(predictions, testing_data[,col]);
   accuracy = sum(sum(result * diag(nclass))) / sum(sum(result));
```

```
    cat("Bootstrapping: mtry=", which.max(acc_param), "ratio = ", ratio, " accuracy
= ", accuracy, ".\n");


    #k-fold cross validation, k = 10
    #Randomly shuffle the data
    acc_param = rep(0, ncol(D));
    training_data<-training_data[sample(nrow(training_data)),]

    #Create 10 equally size folds
    folds <- cut(seq(1,nrow(training_data)),breaks=10,labels=FALSE)


    for (i in 1:10) {
        kfold_testing_indices = which(folds==i,arr.ind=TRUE);
        kfold_testing_data = training_data[kfold_testing_indices,];
        kfold_training_data = training_data[-kfold_testing_indices,];
        for (k in param_set) {
            if (k >= ncol(kfold_training_data)) break;
            rf = randomForest(formula = as.formula(form), data =
kfold_training_data, mtry = k, ntree = 1024);
            predictions = predict(rf, kfold_testing_data);
            result = table(predictions, kfold_testing_data[,col]);
            acc_param[k] = acc_param[k] + sum(sum(result * diag(nclass))) /
sum(sum(result));
        }
    }

    rf = randomForest(formula = as.formula(form), data = training_data, mtry =
which.max(acc_param), ntree = 1024);
    predictions = predict(rf, testing_data);
    result = table(predictions, testing_data[,col]);
    accuracy = sum(sum(result * diag(nclass))) / sum(sum(result));
    cat("K-fold: mtry=", which.max(acc_param), "ratio = ", ratio, " accuracy = ",
accuracy, ".\n");
}
```

TicTacToe.R

```
param_set = c(1, 2, 4, 6, 8, 12, 16, 20, 22, 24, 26, 32, 40, 48, 64);
for (ratio in c(0.1, 0.2, 0.3, 0.4, 0.5)) {
   D = read.csv("tic-tac-toe.csv", header = F);
   nclass = 2;
   col = 'V28';
   form = 'V28 ~ .';
   acc_param = rep(0, ncol(D));
   training_indices = sample(nrow(D), floor(nrow(D) * ratio), replace = FALSE,
prob = NULL);
   training_data <- D[training_indices,];
   testing_data <- D[-training_indices,];

   # Bootstrapping
   for (i in 1:30) {
        bootstrap_indices = sample(length(training_indices), replace = TRUE);
        bootstrap_data = training_data[bootstrap_indices,];
        holdout_data = training_data[-bootstrap_indices,];
      for (k in param_set) {
            if (k >= ncol(bootstrap_data)) break;
            rf = randomForest(formula = as.formula(form), data = bootstrap_data,
mtry = k, ntree = 1024);
            predictions = predict(rf, holdout_data);
            result = table(predictions, holdout_data[,col]);
            acc_param[k] = acc_param[k] + sum(sum(result * diag(nclass))) /
sum(sum(result));
      }
   }
   rf = randomForest(formula = as.formula(form), data = training_data, mtry =
which.max(acc_param), ntree = 1024);
   predictions = predict(rf, testing_data);
   result = table(predictions, testing_data[,col]);
   accuracy = sum(sum(result * diag(nclass))) / sum(sum(result));
   cat("Bootstrapping: mtry=", which.max(acc_param), "ratio = ", ratio, " accuracy
= ", accuracy, ".\n");


   #k-fold cross validation, k = 10
   #Randomly shuffle the data
```

```
    acc_param = rep(0, ncol(D));
    training_data<-training_data[sample(nrow(training_data)),]

    #Create 10 equally size folds
    folds <- cut(seq(1,nrow(training_data)),breaks=10,labels=FALSE)


    for (i in 1:10) {
        kfold_testing_indices = which(folds==i,arr.ind=TRUE);
        kfold_testing_data = training_data[kfold_testing_indices,];
        kfold_training_data = training_data[-kfold_testing_indices,];
        for (k in param_set) {
            if (k >= ncol(kfold_training_data)) break;
            rf = randomForest(formula = as.formula(form), data =
kfold_training_data, mtry = k, ntree = 1024);
            predictions = predict(rf, kfold_testing_data);
            result = table(predictions, kfold_testing_data[,col]);
            acc_param[k] = acc_param[k] + sum(sum(result * diag(nclass))) /
sum(sum(result));
        }
    }

    rf = randomForest(formula = as.formula(form), data = training_data, mtry =
which.max(acc_param), ntree = 1024);
    predictions = predict(rf, testing_data);
    result = table(predictions, testing_data[,col]);
    accuracy = sum(sum(result * diag(nclass))) / sum(sum(result));
    cat("K-fold: mtry=", which.max(acc_param), "ratio = ", ratio, " accuracy = ",
accuracy, ".\n");
}
```

Letter.R

```
param_set = c(1, 2, 4, 6, 8, 12, 16, 20, 22, 24, 26, 32, 40, 48, 64);
for (ratio in c(0.1)) {
   D = read.csv("Letter.csv", header = F);
   nclass = 26;
   col = 'V1';
   form = 'V1 ~ .';
   acc_param = rep(0, ncol(D));
   training_indices = sample(nrow(D), floor(nrow(D) * ratio), replace = FALSE,
prob = NULL);
   training_data <- D[training_indices,];
   testing_data <- D[-training_indices,];

   # Bootstrapping
   for (i in 1:10) {
        bootstrap_indices = sample(length(training_indices), replace = TRUE);
        bootstrap_data = training_data[bootstrap_indices,];
        holdout_data = training_data[-bootstrap_indices,];
      for (k in param_set) {
            if (k >= ncol(bootstrap_data)) break;
            rf = randomForest(formula = as.formula(form), data = bootstrap_data,
mtry = k, ntree = 1024);
            predictions = predict(rf, holdout_data);
            result = table(predictions, holdout_data[,col]);
            acc_param[k] = acc_param[k] + sum(sum(result * diag(nclass))) /
sum(sum(result));
                print(sum(sum(result * diag(nclass))) / sum(sum(result)))
      }
   }
   rf = randomForest(formula = as.formula(form), data = training_data, mtry =
which.max(acc_param), ntree = 1024);
   predictions = predict(rf, testing_data);
   result = table(predictions, testing_data[,col]);
   accuracy = sum(sum(result * diag(nclass))) / sum(sum(result));
   cat("Bootstrapping: mtry=", which.max(acc_param), "ratio = ", ratio, " accuracy
= ", accuracy, ".\n");


   #k-fold cross validation, k = 10
```

```
   #Randomly shuffle the data
   acc_param = rep(0, ncol(D));
   training_data<-training_data[sample(nrow(training_data)),]

   #Create 10 equally size folds
   folds <- cut(seq(1,nrow(training_data)),breaks=10,labels=FALSE)


   for (i in 1:10) {
        kfold_testing_indices = which(folds==i,arr.ind=TRUE);
        kfold_testing_data = training_data[kfold_testing_indices,];
        kfold_training_data = training_data[-kfold_testing_indices,];
        for (k in param_set) {
                if (k >= ncol(kfold_training_data)) break;
                rf = randomForest(formula = as.formula(form), data =
kfold_training_data, mtry = k, ntree = 1024);
                predictions = predict(rf, kfold_testing_data);
                result = table(predictions, kfold_testing_data[,col]);
                acc_param[k] = acc_param[k] + sum(sum(result * diag(nclass))) /
sum(sum(result));
                cat(k, ": ", acc_param[k], "\n");
        }
   }

   rf = randomForest(formula = as.formula(form), data = training_data, mtry =
which.max(acc_param), ntree = 1024);
   predictions = predict(rf, testing_data);
   result = table(predictions, testing_data[,col]);
   accuracy = sum(sum(result * diag(nclass))) / sum(sum(result));
   cat("K-fold: mtry=", which.max(acc_param), "ratio = ", ratio, " accuracy = ",
accuracy, ".\n");
}
```

Magic.R

```
for (ratio in c(0.1, 0.2, 0.3, 0.4, 0.5)) {
   D = read.csv("Magic.csv", header = F);
   nclass = 2;
   col = 'V11';
   form = 'V11 ~ .';
   acc_param = rep(0, ncol(D));
   training_indices = sample(nrow(D), floor(nrow(D) * ratio), replace = FALSE,
prob = NULL);
   training_data <- D[training_indices,];
   testing_data <- D[-training_indices,];

   # Bootstrapping
   for (i in 1:10) {
        bootstrap_indices = sample(length(training_indices), replace = TRUE);
        bootstrap_data = training_data[bootstrap_indices,];
        holdout_data = training_data[-bootstrap_indices,];
       for (k in param_set) {
            if (k >= ncol(bootstrap_data)) break;
            rf = randomForest(formula = as.formula(form), data = bootstrap_data,
mtry = k, ntree = 1024);
            predictions = predict(rf, holdout_data);
            result = table(predictions, holdout_data[,col]);
            acc_param[k] = acc_param[k] + sum(sum(result * diag(nclass))) /
sum(sum(result));
                cat(k, ": ", acc_param[k], "\n");
       }
   }
   rf = randomForest(formula = as.formula(form), data = training_data, mtry =
which.max(acc_param), ntree = 1024);
   predictions = predict(rf, testing_data);
   result = table(predictions, testing_data[,col]);
   accuracy = sum(sum(result * diag(nclass))) / sum(sum(result));
   cat("Bootstrapping: mtry=", which.max(acc_param), "ratio = ", ratio, " accuracy
= ", accuracy, ".\n");


   #k-fold cross validation, k = 10
   #Randomly shuffle the data
```

```
    acc_param = rep(0, ncol(D));
    training_data<-training_data[sample(nrow(training_data)),]

    #Create 10 equally size folds
    folds <- cut(seq(1,nrow(training_data)),breaks=10,labels=FALSE)


    for (i in 1:10) {
        kfold_testing_indices = which(folds==i,arr.ind=TRUE);
        kfold_testing_data = training_data[kfold_testing_indices,];
        kfold_training_data = training_data[-kfold_testing_indices,];
        for (k in param_set) {
            if (k >= ncol(kfold_training_data)) break;
            rf = randomForest(formula = as.formula(form), data =
kfold_training_data, mtry = k, ntree = 1024);
            predictions = predict(rf, kfold_testing_data);
            result = table(predictions, kfold_testing_data[,col]);
            acc_param[k] = acc_param[k] + sum(sum(result * diag(nclass))) /
sum(sum(result));
            cat(k, ": ", acc_param[k], "\n");
        }
    }

    rf = randomForest(formula = as.formula(form), data = training_data, mtry =
which.max(acc_param), ntree = 1024);
    predictions = predict(rf, testing_data);
    result = table(predictions, testing_data[,col]);
    accuracy = sum(sum(result * diag(nclass))) / sum(sum(result));
    cat("K-fold: mtry=", which.max(acc_param), "ratio = ", ratio, " accuracy = ",
accuracy, ".\n");
}
```