LevWilliamsRF

Bagging Function

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 3.4.4
```

```
## -- Attaching packages ---------------------------------- tidyverse 1.2.1 --
```

```
## v ggplot2 3.1.0     v purrr   0.2.5
## v tibble  1.4.2     v dplyr   0.7.8
## v tidyr   0.8.1     v stringr 1.3.1
## v readr   1.1.1     v forcats 0.3.0
```

```
## Warning: package 'ggplot2' was built under R version 3.4.4
```

```
## Warning: package 'tibble' was built under R version 3.4.4
```

```
## Warning: package 'tidyr' was built under R version 3.4.4
```

```
## Warning: package 'readr' was built under R version 3.4.4
```

```
## Warning: package 'purrr' was built under R version 3.4.4
```

```
## Warning: package 'dplyr' was built under R version 3.4.4
```

```
## Warning: package 'stringr' was built under R version 3.4.4
```

```
## Warning: package 'forcats' was built under R version 3.4.4
```

```
## -- Conflicts ------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

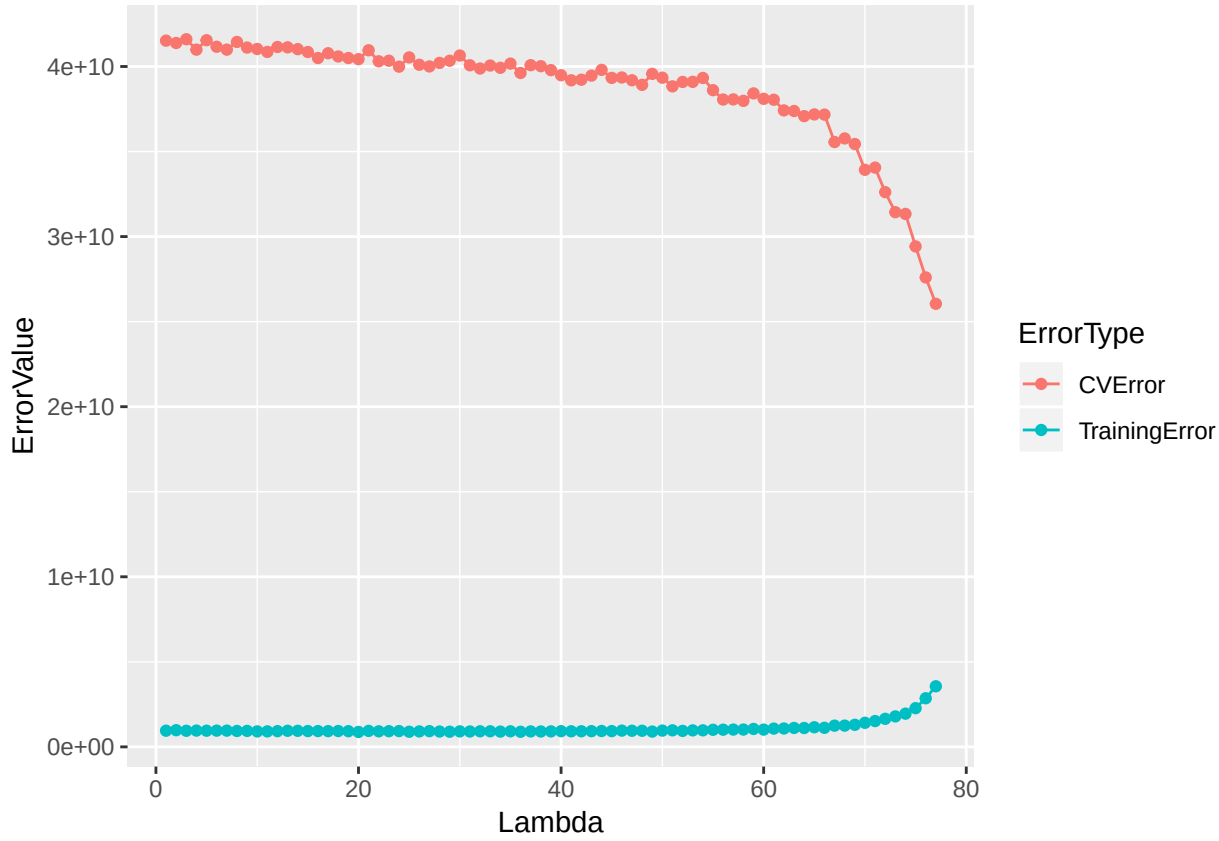
```
library(ggplot2)
library(rpart)

library(parallel)
ModelBagger <- function(formulastr, trained, learner,B,Tree = FALSE,m = 1000, cores = 1){


  # Calculate the number of cores
  no_cores <- cores

  # Initiate cluster
  cl <- makeCluster(no_cores)
  library(rpart)
  clusterExport(cl, "rpart")

  DataBoot <- function(data){
  threshold <- ceiling(.632*nrow(data))
  samp <- sample(1:nrow(data),threshold, replace = TRUE)
  return(data[samp,])
}

DataTreeBoot <- function(data,m){
  threshold <- ceiling(.632*nrow(data))
  samp <- sample(1:nrow(data),threshold, replace = TRUE)
```

```
  sampcol <- sample(2:ncol(data),m)
  return(data[samp,-sampcol])
}



  if((m - 1) > length(2:ncol(trained))){p
    m - 2 = length(2:ncol(trained))
  }



  SubSamper <- function(vect){
    return(DataBoot(trained))
  }

  TreeSamper <- function(vect){
    return(DataTreeBoot(trained,m))
  }

  ModelApplier <- function(data){
    return(learner(formulastr, data))
  }

  TreeApplier <- function(data){
    return(learner(formulastr, data, method = "anova"))
  }

  if(Tree == FALSE){
  BootData <- parLapply(cl,1:B,SubSamper)
  mlist <- parLappy(cl,BootData,ModelApplier)
  }

  else{
  BootData <- parLapply(cl,1:B,TreeSamper)
  mlist <- parLapply(cl,BootData,TreeApplier)
  }
  stopCluster(cl)
  return(mlist)
}


BagPredict <- function(bagm,tested,predictionfunc){

  applyfunc <- function(temp){
      values <- predictionfunc(temp, newdata = tested)
      return(values)
  }
  OutputVecs <- lapply(bagm, applyfunc)
  OutputM <- do.call(cbind,OutputVecs)
  #ystar   <- apply(OutputM, 1, trimean)
  ystar <- rowMeans(OutputM)
  return(ystar)
```

```
}

BagError <- function(yreal,yhat){
  MSEBag <- sum((yhat-yreal)^2)/length(yhat)
  return(MSEBag)
}

library(ggplot2)
CrossVal <- function(ModelString, Training, CrossValidation, Learner, Tree = FALSE,B,TuningParam,TrainR
{
  TrainingError <- numeric(TuningParam)
  CVError <- numeric(TuningParam)
  Lambda <- 1:TuningParam
  for(i in 1:TuningParam){

    Ensemb <- ModelBagger(ModelString,Training,Learner,B,Tree,i)
    PredTrain <- BagPredict(Ensemb,Training,predict)
    PredCV <- BagPredict(Ensemb,CrossValidation,predict)
    TrainingError[i] <- BagError(TrainResp,PredTrain)
    CVError[i] <- BagError(TrainResp,PredCV)
  }
  LearningCurve <- cbind(TrainingError,CVError,Lambda)
  return(LearningCurve)
}
```

KaggleComp

```
HousingTrain <- read.csv("C:/Users/LardR/Desktop/KaggleSuessProject/train.csv")
HousingTest <- read.csv("C:/Users/LardR/Desktop/KaggleSuessProject/test.csv")
SwapVec <- c(81,2:80,1)
HousingTrain <- HousingTrain[,SwapVec]
HousingTrain <- HousingTrain[-81]
TrainRows <- sample(1:nrow(HousingTrain),ceiling(.8*nrow(HousingTrain)))
TrainDHouse <- HousingTrain[TrainRows,]
CVDHouse <- HousingTrain[-TrainRows,]
HouseSellPrice <- TrainDHouse[,1]
Diag <- CrossVal(SalePrice ~ .,Training = TrainDHouse,CrossValidation = CVDHouse, rpart, Tree=TRUE, 100


Diag <- as.data.frame(Diag)
GathDiag <- gather(Diag, key = ErrorType, value = ErrorValue, -Lambda)
ggplot(GathDiag,aes(x = Lambda, y = ErrorValue, color = ErrorType)) + geom_point() + geom_line()
```




```
testreord <- c(2:80,1)
HousingTest <- HousingTest[,testreord]
Id <- HousingTest[,80]
HousingTest <- HousingTest[,-80]
Bag <- ModelBagger(SalePrice ~ ., HousingTrain, rpart, 5000, Tree = TRUE,m = 60, cores = 1)
SalePrice <- BagPredict(Bag,HousingTest,predict)


KaggleSubmis <- cbind(Id,SalePrice)
KaggleSubmis <- as.data.frame(KaggleSubmis)
write.csv(KaggleSubmis, "KaggleSubmis10.csv")

#library(randomForest)
#?randomForest

#DefaultRF <- randomForest(SalePrice ~ ., HousingTrain, ntree = 10000, na.action = #na.roughfix)
#ReponseVals <- predict(DefaultRF,HousingTest).
```