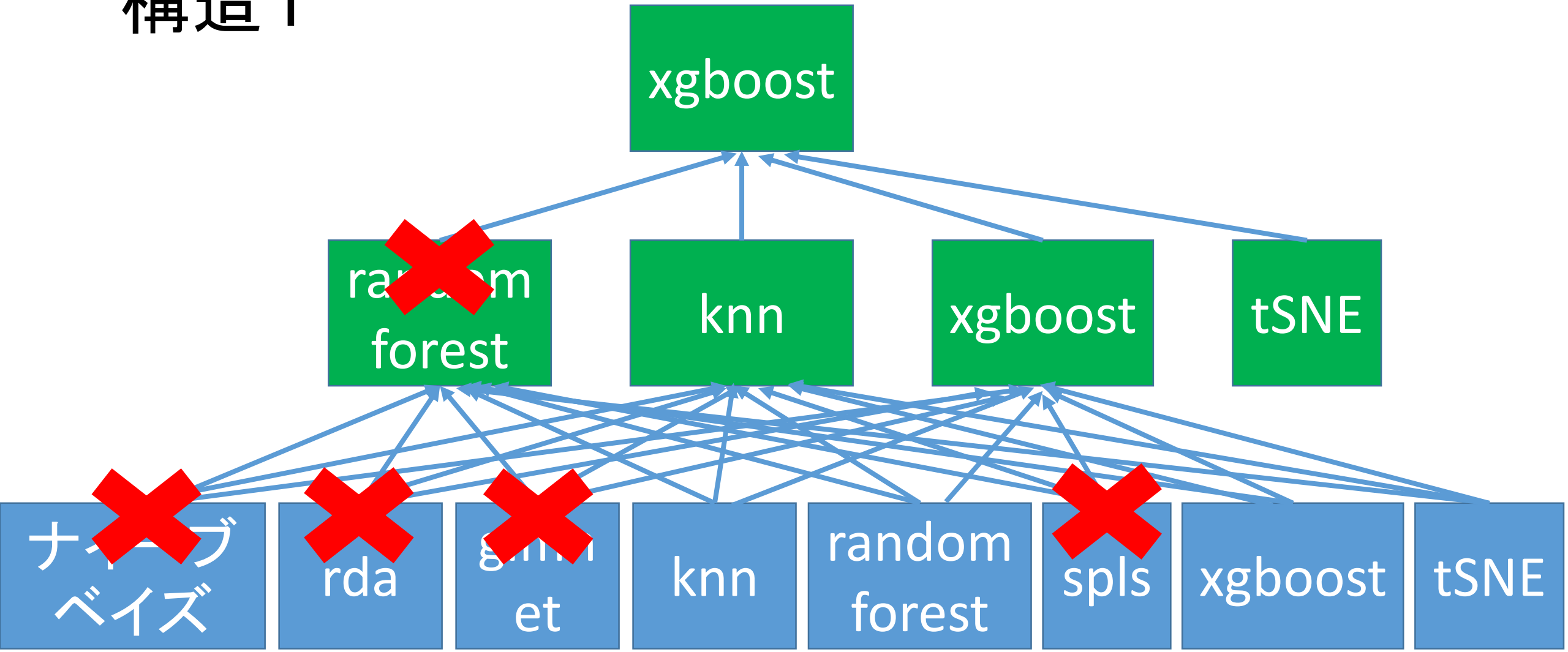
Stacking
多項分類編

鈴木瑞人
東京大学大学院　新領域創成科学研究科
メディカル情報生命専攻
博士課程1年

構造1


xgboost
random forest
knn
xgboost
tSNE
ナイーブベイズ
rda
glmnet
knn
random forest
spls
xgboost
tSNE

構造1を作成する。

- (データを学習用とテスト用に分割する。(今回は分割しない))
- 今回はすべてのデータを学習用とする
- まずは最下層の学習器を作成し、テストデータをもとに、予測値を算出する。
- テスト用データで、第2層の学習器を作成する。説明変数は最下層から上がってきた予測値。

並列計算環境を整える

- AWS、WindowsServer2016, m4.xlarge
- Total Memory:16GB
- Architechture: AMD64

並列処理

```
install.packages("doParallel", quiet = TRUE, dependencies=T)
library(doParallel)
detectCores()
Cl <- makeCluster(detectCores())
registerDoParallel(Cl)
#並列処理を終えるとき(まだ実行しない)
stopCluster(Cl)
```

```
> detectCores()
[1] 4
```

RはデフォルトではCPUをシングルコアしか使わないので、detectCores()関数で、コアの数を把握し、makeCluster関数とregisterDoParallel関数で、コアの数だけCPUを使用する(筆者が試したのは2コアまで)。

```
getwd()
setwd("C:/Users/Administrator/Documents/leaf")
dir()
data1=read.csv("train.csv", header=T)
dim(data1)
```

```
> dim(data1)
[1] 990 194
```

Idはいらないので消去

data1=data1[,2:194]

str(data1)

```
> data1=read.csv("train.csv", header=T)
> data1=data1[,2:194]
> str(data1)
'data.frame':   990 obs. of  193 variables:
 $ species  : Factor w/ 99 levels "Acer_Capillipes",..: 4 50 66
 $ margin1  : num  0.00781 0.00586 0.00586 0 0.00586 ...
 $ margin2  : num  0.02344 0 0.00977 0.00391 0.00391 ...
 $ margin3  : num  0.0234 0.0312 0.0195 0.0234 0.0488 ...
 $ margin4  : num  0.00391 0.01562 0.00781 0.00586 0.00977 ...
 $ margin5  : num  0.01172 0.02539 0.00391 0.02148 0.01367 ...
 $ margin6  : num  0.00977 0.00195 0.00586 0.01953 0.01562 ...
 $ margin7  : num  0.02734 0.01953 0.06836 0.02344 0.00586 ...
 $ margin8  : num  0 0 0 0 0 0 0 0 0 0 ...
 $ margin9  : num  0.00195 0 0 0.01367 0 ...
 $ margin10 : num  0.0332 0.00781 0.04492 0.01758 0.00586 ...
 $ margin11 : num  0.01367 0.00391 0.00781 0.00195 0.00195 ...
 $ margin12 : num  0.0195 0.0273 0.0117 0.0195 0.0449 ...
 $ margin13 : num  0.06641 0.02344 0.02148 0.00195 0.04102 ...
 $ margin14 : num  0 0 0.00195 0.00391 0.01172 ...
 $ margin15 : num  0.0293 0.0332 0.0254 0.0352 0.041 ...
 $ margin16 : num  0 0 0 0 0 0 0 0 0 0 ...
 $ margin17 : num  0.03125 0.00977 0.00977 0.00586 0.00977 ...
 $ margin18 : num  0.01172 0.00977 0.01172 0 0.01562 ...
 $ margin19 : num  0 0.00781 0.00781 0.00195 0.01172 ...
 $ margin20 : num  0.02539 0.00781 0.00586 0.00391 0.00781 ...
```

- 1列目に目的変数、2列目以降に説明変数。

学習用データ

X=data1[,2:193]

Y=data1[,1]

```
install.packages("caret", quiet=T, dependencies=T)
library(caret)
```

randomForest

```
fitControl = trainControl(method="LGOCV", p = 0.80, repeats=1)
```

randomForestでのモデル作成

```
rf.model = train(
                x = X,
                y = Y,
                 method = "rf",
                 trControl = fitControl,
                 tuneLength=10,
                 preProc = c("center", "scale")
                 )
```

```
> rf.model
Random Forest

990 samples
192 predictors
 99 classes: 'Acer_Capillipes', 'Acer_Circinatum', 'Acer_Mono', 'Acer_Opalus',

Pre-processing: centered (192), scaled (192)
Resampling: Repeated Train/Test Splits Estimated (25 reps, 80%)
Summary of sample sizes: 792, 792, 792, 792, 792, 792, ...
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
    2   0.9771717  0.9769388
   23   0.9852525  0.9851020
   44   0.9820202  0.9818367
   65   0.9763636  0.9761224
   86   0.9698990  0.9695918
  107   0.9672727  0.9669388
  128   0.9648485  0.9644898
  149   0.9624242  0.9620408
  170   0.9622222  0.9618367
  192   0.9579798  0.9575510

Accuracy was used to select the optimal model using  the largest value.
The final value used for the model was mtry = 23.
```

randomForestでの予測値算出

```
rf.model
rf.pred=predict(rf.model, X, type="prob")
rf.pred
write.csv(rf.pred, "rf.csv", row.names=F)
rf.result=read.csv("rf.csv", header=T)
str(rf.result)
```

```
> write.csv(rf.pred, "rf.csv", row.names=F)
> rf.result=read.csv("rf.csv", header=T)
> str(rf.result)
'data.frame':   990 obs. of  99 variables:
 $ Acer_Capillipes              : num  0.014 0.03 0.018 0 0.004 0 0 0 0.002 0 ..
 $ Acer_Circinatum              : num  0 0 0 0 0.018 0 0 0 0 0 ...
 $ Acer_Mono                    : num  0 0 0 0.004 0 0.014 0 0 0 0.004 ...
 $ Acer_Opalus                  : num  0.81 0.004 0 0.008 0.002 0 0 0 0 0 ...
 $ Acer_Palmatum                : num  0 0 0 0 0.002 0 0 0 0 0 ...
 $ Acer_Pictum                  : num  0 0 0 0 0.006 0 0.008 0.008 0 0 ...
 $ Acer_Platanoids              : num  0 0 0 0 0 0 0 0.002 0 0 ...
 $ Acer_Rubrum                  : num  0.002 0.008 0 0.006 0.002 0 0 0 0.006 0.(
 $ Acer_Rufinerve               : num  0 0.016 0 0.002 0 0 0 0 0 0.002 ...
 $ Acer_Saccharinum             : num  0 0 0.002 0 0.006 0 0.004 0.006 0 0 ...
 $ Alnus_Cordata                : num  0 0.004 0.002 0 0.002 0 0.004 0 0 0 ...
 $ Alnus_Maximowiczii           : num  0 0.016 0 0 0 0 0 0 0 0 ...
 $ Alnus_Rubra                  : num  0.006 0.004 0.002 0.002 0 0 0 0 0.016 0 .
 $ Alnus_Sieboldiana            : num  0 0.002 0.004 0.002 0 0 0 0 0.004 0 ...
 $ Alnus_Viridis                : num  0 0.006 0 0 0 0 0.002 0.002 0 0.002 ...
 $ Arundinaria_Simonii          : num  0 0 0 0 0 0 0 0 0 0.002 ...
 $ Betula_Austrosinensis        : num  0 0.002 0 0 0.016 0 0.002 0.004 0 0 ...
 $ Betula_Pendula               : num  0.004 0.002 0 0.008 0.002 0 0 0 0 0.002 .
 $ Callicarpa_Bodinieri         : num  0 0.01 0.002 0 0 0 0 0 0.006 0 ...
```

モデルを保存する

#モデルを保存する(こうするとカレントディレクトリにrf1というファイルができる)

saveRDS(rf.model, file="rf1")

#読み込むときは以下のようにして読み込む

model.read <- readRDS(file="rf1")

```
> model.read
Random Forest

990 samples
192 predictors
 99 classes: 'Acer_Capillipes', 'Acer_Circinatum', 'Acer_Mono', 'Acer_Opalus',

Pre-processing: centered (192), scaled (192)
Resampling: Repeated Train/Test Splits Estimated (25 reps, 80%)
Summary of sample sizes: 792, 792, 792, 792, 792, 792, ...
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
    2   0.9771717  0.9769388
   23   0.9852525  0.9851020
   44   0.9820202  0.9818367
   65   0.9763636  0.9761224
   86   0.9698990  0.9695918
  107   0.9672727  0.9669388
  128   0.9648485  0.9644898
  149   0.9624242  0.9620408
  170   0.9622222  0.9618367
  192   0.9579798  0.9575510

Accuracy was used to select the optimal model using  the largest value.
The final value used for the model was mtry = 23.
```

knn

```
fitControl = trainControl(method="LGOCV", p = 0.80, repeats=1)
```

knnでのモデル作成

```
knn.model = train(
                  x = X,
                  y = Y,
                   method = "knn",
                   trControl = fitControl,
                   tuneLength=10,
                   preProc = c("center", "scale")
                   )
```

```
> knn.model
k-Nearest Neighbors

990 samples
192 predictors
 99 classes: 'Acer_Capillipes', 'Acer_Circinatum', 'Acer_Mono', 'Acer_Opalus',

Pre-processing: centered (192), scaled (192)
Resampling: Repeated Train/Test Splits Estimated (25 reps, 80%)
Summary of sample sizes: 792, 792, 792, 792, 792, 792, ...
Resampling results across tuning parameters:

  k   Accuracy   Kappa
   5  0.9692929  0.9689796
   7  0.9597980  0.9593878
   9  0.9442424  0.9436735
  11  0.9335354  0.9328571
  13  0.9193939  0.9185714
  15  0.9070707  0.9061224
  17  0.8915152  0.8904082
  19  0.8674747  0.8661224
  21  0.8414141  0.8397959
  23  0.8197980  0.8179592

Accuracy was used to select the optimal model using  the largest value.
The final value used for the model was k = 5.
```

モデルを保存する

#モデルを保存する(こうするとカレントディレクトリにrf1というファイルができる)

saveRDS(knn.model, file="knn1")

#読み込むときは以下のようにして読み込む

model.read <- readRDS(file="knn1")

```
> model.read
k-Nearest Neighbors

990 samples
192 predictors
 99 classes: 'Acer_Capillipes', 'Acer_Circinatum', 'Acer_Mono', 'Acer_Opalus',

Pre-processing: centered (192), scaled (192)
Resampling: Repeated Train/Test Splits Estimated (25 reps, 80%)
Summary of sample sizes: 792, 792, 792, 792, 792, 792, ...
Resampling results across tuning parameters:

  k   Accuracy   Kappa
   5  0.9692929  0.9689796
   7  0.9597980  0.9593878
   9  0.9442424  0.9436735
  11  0.9335354  0.9328571
  13  0.9193939  0.9185714
  15  0.9070707  0.9061224
  17  0.8915152  0.8904082
  19  0.8674747  0.8661224
  21  0.8414141  0.8397959
  23  0.8197980  0.8179592

Accuracy was used to select the optimal model using  the largest value.
The final value used for the model was k = 5.
  '
```

knnでの予測値算出

```
knn.model
knn.pred=predict(knn.model, X, type="prob")
knn.pred
write.csv(knn.pred, "knn.csv", row.names=F)
knn.result=read.csv("knn.csv", header=T)
str(knn.result)
```

```
> str(knn.result)
'data.frame':   990 obs. of  99 variables:
 $ Acer_Capillipes               : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Acer_Circinatum               : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Acer_Mono                     : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Acer_Opalus                   : num  1 0 0 0 0 0 0 0 0 0 ...
 $ Acer_Palmatum                 : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Acer_Pictum                   : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Acer_Platanoids               : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Acer_Rubrum                   : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Acer_Rufinerve                : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Acer_Saccharinum              : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Alnus_Cordata                 : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Alnus_Maximowiczii            : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Alnus_Rubra                   : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Alnus_Sieboldiana             : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Alnus_Viridis                 : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Arundinaria_Simonii           : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Betula_Austrosinensis         : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Betula_Pendula                : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Callicarpa_Bodinieri          : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Castanea_Sativa               : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Celtis_Koraiensis             : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Cercis_Siliquastrum           : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Cornus_Chinensis              : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Cornus_Controversa            : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Cornus_Macrophylla            : num  0 0 0 0 0 0 0 0 0 0 ...
```

Xgboostは以前と同じ作り方

学習用データの目的変数をxgboost用に加工

```
# 目的変数
y <- Y
#Factor型である目的変数を、0,1,2,3,,,98, の整数にする。
#xgboost で既定されいるクラスは 0 baseなため。
y <- as.integer(y)-1
```

学習用データの説明変数をxgboost用に加工

```
#xgboostパッケージのダウンロード・インストール
install.packages("xgboost",dependencies=T)
library(xgboost)
# xgboost を使うときのため行列に変換
x <- as.matrix(X)
```

パラメータ指定

```
set.seed(123) # 固定シードで試す
param <- list("objective" = "multi:softprob",
              # 多クラスの分類で各クラスに所属する確率を求める
              "eval_metric" = "mlogloss", # 損失関数の設定
               "num_class" = 99 # class が今回は99個存在。
              )
```

予測をする前に最適な木の数を探す

```
#スタージェスの公式
k<-round(1+log2(nrow(X)))
#適当な数を入れる
cv.nround <- 50 #search
#nroundsを決めるためのモデル作成
bst.cv <- xgb.cv(param=param, data = x, label = y,  nfold = k,
nrounds=cv.nround)
```

```
[26]    train-mlogloss:0.055940+0.000294        test-mlogloss:0.819161+0.075457
[27]    train-mlogloss:0.055738+0.000302        test-mlogloss:0.818733+0.075528
[28]    train-mlogloss:0.055605+0.000312        test-mlogloss:0.818417+0.075511
[29]    train-mlogloss:0.055514+0.000322        test-mlogloss:0.818313+0.075783
[30]    train-mlogloss:0.055452+0.000329        test-mlogloss:0.818177+0.075924
[31]    train-mlogloss:0.055403+0.000337        test-mlogloss:0.818085+0.076030
[32]    train-mlogloss:0.055362+0.000339        test-mlogloss:0.818020+0.076119
[33]    train-mlogloss:0.055334+0.000333        test-mlogloss:0.817976+0.076177
[34]    train-mlogloss:0.055320+0.000330        test-mlogloss:0.817966+0.076239
[35]    train-mlogloss:0.055311+0.000329        test-mlogloss:0.817966+0.076290
[36]    train-mlogloss:0.055305+0.000328        test-mlogloss:0.817974+0.076344
[37]    train-mlogloss:0.055301+0.000327        test-mlogloss:0.817981+0.076406
[38]    train-mlogloss:0.055300+0.000327        test-mlogloss:0.818014+0.076425
[39]    train-mlogloss:0.055299+0.000327        test-mlogloss:0.818046+0.076443
[40]    train-mlogloss:0.055298+0.000327        test-mlogloss:0.818075+0.076459
[41]    train-mlogloss:0.055297+0.000327        test-mlogloss:0.818102+0.076474
[42]    train-mlogloss:0.055296+0.000326        test-mlogloss:0.818127+0.076487
[43]    train-mlogloss:0.055296+0.000327        test-mlogloss:0.818150+0.076500
[44]    train-mlogloss:0.055296+0.000326        test-mlogloss:0.818172+0.076512
[45]    train-mlogloss:0.055295+0.000326        test-mlogloss:0.818192+0.076523
[46]    train-mlogloss:0.055295+0.000326        test-mlogloss:0.818211+0.076533
[47]    train-mlogloss:0.055295+0.000326        test-mlogloss:0.818228+0.076542
[48]    train-mlogloss:0.055294+0.000326        test-mlogloss:0.818244+0.076551
[49]    train-mlogloss:0.055294+0.000326        test-mlogloss:0.818259+0.076559
```

```
set.seed(123)
nround <- 35
# モデルの構築
bst <- xgboost(param=param, data = x, label = y, nrounds=nround)
```

モデルを保存する

#モデルを保存する(こうするとカレントディレクトリにrf1というファイルができる)

saveRDS(bst, file="bst1")

#読み込むときは以下のようにして読み込む

model.read <- readRDS(file="bst1")

test=as.matrix(X)

pred <- predict(bst,test)

head(pred)

str(pred)

```
> test=as.matrix(test)
> pred <- predict(bst,test[,-1])
> head(pred)
[1] 0.008701433 0.003948880 0.002994578 0.042810339 0.004569929 0.004519264
> str(pred)
 num [1:58806] 0.0087 0.00395 0.00299 0.04281 0.00457 ...
```

一列のベクトル

#99列の行列にする。数字の配列は左から右

q=matrix(pred,ncol=99, byrow=T)

new=data.frame(q)　#データフレーム型にする。

列名を付ける。

#列順は、きっとABC順(sample_submission.csvも同じ)
#訓練データの植物の名前をとってくる。
y <- Y
#重複をunique関数で取り除き、Factor型を整数型にする。
z=as.integer(unique(y))

列名を付ける。

#植物名とFactorとしての数字をペアにする(数字が実体で名前が植物名) 。

names(z)=unique(y)

z

列名を付ける。

#数字でソートする(昇順)

z1=sort(z)

列名を付ける。

```
#列に名前を付ける。
colnames(new)=names(z1)
head(new)
```

```
[1] "xgb.Booster"
> test=as.matrix(X)
> pred <- predict(bst,test)
> head(pred)
[1] 0.0010087815 0.0005941767 0.0004505859 0.9348506331 0.0006876242
[6] 0.0005075474
> q=matrix(pred,ncol=99, byrow=T)
> new=data.frame(q)
> y <- Y
> z=as.integer(unique(y))
>   names(z)=unique(y)
> z1=sort(z)
> colnames(new)=names(z1)
> head(new)
  Acer_Capillipes Acer_Circinatum    Acer_Mono  Acer_Opalus Acer_Palmatum
1    0.0010087815    0.0005941767 0.0004505859 0.9348506331  0.0006876242
2    0.0003677055    0.0001870920 0.0002047331 0.0002867334  0.0002165164
3    0.0041927136    0.0003917375 0.0006995361 0.0009695545  0.0003865193
4    0.0005841982    0.0002713962 0.0003690519 0.0002778295  0.0003140793
5    0.0009656262    0.0003714889 0.0003364001 0.0002848516  0.0003665402
6    0.0007175673    0.0010185469 0.0066869641 0.0007595062  0.0010049792
   Acer_Pictum Acer_Platanoids  Acer_Rubrum Acer_Rufinerve Acer_Saccharinum
1 0.0005075474    0.0006095465 0.0012825958   0.0005635499     0.0006177971
2 0.0001970436    0.0002164821 0.0001662115   0.0002195815     0.0001945295
3 0.0005126085    0.0003864581 0.0004404671   0.0002833574     0.0003472689
```

ちなみに以上をまとめると

```
test=as.matrix(X)
pred <- predict(bst,test)
head(pred)
q=matrix(pred,ncol=99, byrow=T)
new=data.frame(q)
y <- Y
z=as.integer(unique(y))
names(z)=unique(y)
z1=sort(z)
colnames(new)=names(z1)   #ここで名前付けた
head(new) #中身確認
```

今のデータをcsvファイルとして出力

write.csv(new,"xgboost.csv", row.names=F)

t-SNEの特徴量の作り方

tsneパッケージ

```
install.packages("tsne",dependencies=T)
library(tsne)
```

```
tsne.X = tsne(X)
head(tsne.X)
write.csv(tsne.X, "tsne1.csv",row.names=F)
head(read.csv("tsne1.csv",header=T))
```

```
> library(tsne)
> tsne.X = tsne(X)
sigma summary: Min. : 0.4501 |1st Qu. : 0.6781 |Median : 0.7394 |Mean : 0.7536
Epoch: Iteration #100 error is: 16.8330375209124
Epoch: Iteration #200 error is: 0.710906116832207
Epoch: Iteration #300 error is: 0.567923644194328
Epoch: Iteration #400 error is: 0.522121163989188
Epoch: Iteration #500 error is: 0.508759364915291
Epoch: Iteration #600 error is: 0.502690857852021
Epoch: Iteration #700 error is: 0.499225539741033
Epoch: Iteration #800 error is: 0.496853197243904
Epoch: Iteration #900 error is: 0.49360074506281 5
Epoch: Iteration #1000 error is: 0.491145630685748
```

```
> head(tsne.X)
             [,1]        [,2]
[1,] -11.454289  14.950662
[2,]  22.874065  17.639580
[3,]   4.154852 -31.425611
[4,]  -1.408921 -16.627144
[5,] -58.165927  -4.156737
[6,] -49.816635 -26.391074
```

```
> head(read.csv("tsne1.csv",header=T))
          V1          V2
1 -11.454289  14.950662
2  22.874065  17.639580
3   4.154852 -31.425611
4  -1.408921 -16.627144
5 -58.165927  -4.156737
6 -49.816635 -26.391074
```

2層目の学習器の作り方

- 説明変数を作る。
- Randomforestによる学習
- Knnによる学習
- Xgboostによる学習

説明変数を作る

```
rf.result=read.csv("rf.csv", header=T)
str(rf.result)
new.X=rf.result
```

```
> rf.result=read.csv("rf.csv", header=T)
> str(rf.result)
'data.frame':   990 obs. of  99 variables:
 $ Acer_Capillipes              : num  0.014 0.03 0.018 0 0.004 0 0 0 0.002 0 ...
 $ Acer_Circinatum              : num  0 0 0 0 0.018 0 0 0 0 0 ...
 $ Acer_Mono                    : num  0 0 0 0.004 0 0.014 0 0 0 0.004 ...
 $ Acer_Opalus                  : num  0.81 0.004 0 0.008 0.002 0 0 0 0 0 ...
 $ Acer_Palmatum                : num  0 0 0 0 0.002 0 0 0 0 0 ...
 $ Acer_Pictum                  : num  0 0 0 0 0.006 0 0.008 0.008 0 0 ...
 $ Acer_Platanoids              : num  0 0 0 0 0 0 0 0.002 0 0 ...
 $ Acer_Rubrum                  : num  0.002 0.008 0 0.006 0.002 0 0 0 0.006 0.0$
 $ Acer_Rufinerve               : num  0 0.016 0 0.002 0 0 0 0 0 0.002 ...
 $ Acer_Saccharinum             : num  0 0 0.002 0 0.006 0 0.004 0.006 0 0 ...
 $ Alnus_Cordata                : num  0 0.004 0.002 0 0.002 0 0.004 0 0 0 ...
 $ Alnus_Maximowiczii           : num  0 0.016 0 0 0 0 0 0 0 0 ...
 $ Alnus_Rubra                  : num  0.006 0.004 0.002 0.002 0 0 0 0 0.016 0 .$
 $ Alnus_Sieboldiana            : num  0 0.002 0.004 0.002 0 0 0 0 0.004 0 ...
 $ Alnus_Viridis                : num  0 0.006 0 0 0 0 0.002 0.002 0 0.002 ...
 $ Arundinaria_Simonii          : num  0 0 0 0 0 0 0 0 0 0.002 ...
 $ Betula_Austrosinensis        : num  0 0.002 0 0 0.016 0 0.002 0.004 0 0 ...
```

説明変数を作る

knn.result=read.csv("knn.csv", header=T)

str(knn.result)

new.X=cbind(new.X,knn.result)

dim(new.X)

```
> new.X=cbind(new.X,knn.result)
> dim(new.X)
[1] 990 198
```

```
> knn.result=read.csv("knn.csv", header=T)
> str(knn.result)
'data.frame':   990 obs. of  99 variables:
 $ Acer_Capillipes              : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Acer_Circinatum              : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Acer_Mono                    : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Acer_Opalus                  : num  1 0 0 0 0 0 0 0 0 0 ...
 $ Acer_Palmatum                : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Acer_Pictum                  : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Acer_Platanoids              : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Acer_Rubrum                  : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Acer_Rufinerve               : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Acer_Saccharinum             : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Alnus_Cordata                : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Alnus_Maximowiczii           : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Alnus_Rubra                  : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Alnus_Sieboldiana            : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Alnus_Viridis                : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Arundinaria_Simonii          : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Betula_Austrosinensis        : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Betula_Pendula               : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Callicarpa_Bodinieri         : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Castanea_Sativa              : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Celtis_Koraiensis            : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Cercis_Siliquastrum          : num  0 0 0 0 0 0 0 0 0 0 ...
```

説明変数を作る

```
xgboost.result=read.csv("xgboost.csv", header=T)
str(xgboost.result)
new.X=cbind(new.X, xgboost.result)
dim(new.X)
tsne1.result=read.csv("tsne1.csv", header=T)
str(tsne1.result)
new.X=cbind(new.X, tsne1.result)
dim(new.X)
#説明変数の名前を変える
X=new.X
#説明変数1の完成版を出力
write.csv(new.X,"explain.csv",row.names=
head(read.csv("explain.csv",header=T))
```

```
> new.X=cbind(new.X, xgboost.result)
> dim(new.X)
[1] 990 297
> tsne1.result=read.csv("tsne1.csv", header=T)
> str(tsne1.result)
'data.frame':   990 obs. of  2 variables:
 $ V1: num  -11.45 22.87 4.15 -1.41 -58.17 ...
 $ V2: num  14.95 17.64 -31.43 -16.63 -4.16 ...
> new.X=cbind(new.X, tsne1.result)
> dim(new.X)
[1] 990 299
```

```
> xgboost.result=read.csv("xgboost.csv", header=T)
> str(xgboost.result)
'data.frame':   990 obs. of  99 variables:
 $ Acer_Capillipes                : num  0.001009 0.000368 0.004193 0.000584 0.000$
 $ Acer_Circinatum                : num  0.000594 0.000187 0.000392 0.000271 0.000$
 $ Acer_Mono                      : num  0.000451 0.000205 0.0007 0.000369 0.00033$
 $ Acer_Opalus                    : num  0.934851 0.000287 0.00097 0.000278 0.0002$
 $ Acer_Palmatum                  : num  0.000688 0.000217 0.000387 0.000314 0.000$
 $ Acer_Pictum                    : num  0.000508 0.000197 0.000513 0.000394 0.000$
 $ Acer_Platanoids                : num  0.00061 0.000216 0.000386 0.000358 0.0003$
 $ Acer_Rubrum                    : num  0.001283 0.000166 0.00044 0.000628 0.0008$
 $ Acer_Rufinerve                 : num  0.000564 0.00022 0.000283 0.000417 0.0003$
 $ Acer_Saccharinum               : num  0.000618 0.000195 0.000347 0.000282 0.000$
 $ Alnus_Cordata                  : num  0.001293 0.004624 0.000593 0.000968 0.000$
 $ Alnus_Maximowiczii             : num  0.000528 0.000275 0.000297 0.00035 0.0002$
 $ Alnus_Rubra                    : num  0.000667 0.000188 0.001075 0.000688 0.000$
 $ Alnus_Sieboldiana              : num  0.000562 0.000177 0.000316 0.001115 0.001$
 $ Alnus_Viridis                  : num  0.000567 0.000178 0.000318 0.000259 0.000$
 $ Arundinaria_Simonii            : num  0.000737 0.000232 0.000414 0.000337 0.000$
```

```
> write.csv(new.X,"explain.csv",row.names=F)
> head(read.csv("explain.csv",header=T))
  Acer_Capillipes Acer_Circinatum Acer_Mono Acer_Opalus Acer_Palmatum
1           0.014           0.000     0.000       0.810          0.000
2           0.030           0.000     0.000       0.004          0.000
3           0.018           0.000     0.000       0.000          0.000
4           0.000           0.000     0.004       0.008          0.000
5           0.004           0.018     0.000       0.002          0.002
6           0.000           0.000     0.014       0.000          0.000
  Acer_Pictum Acer_Platanoids Acer_Rubrum Acer_Rufinerve Acer_Saccharinum
1       0.000               0       0.002          0.000             0.000
2       0.000               0       0.008          0.016             0.000
3       0.000               0       0.000          0.000             0.002
4       0.000               0       0.006          0.002             0.000
5       0.006               0       0.002          0.000             0.006
6       0.000               0       0.000          0.000             0.000
  Alnus_Cordata Alnus_Maximowiczii Alnus_Rubra Alnus_Sieboldiana Alnus_Viridis
1         0.000              0.000       0.006             0.000         0.000
2         0.004              0.016       0.004             0.002         0.006
```

第2層のknnでの学習

```
fitControl = trainControl(method="LGOCV", p = 0.80, repeats=1)
```

第2層のknnでのモデル作成

```
knn2.model = train(
                x = X,
                y = Y,
                 method = "knn",
                 trControl = fitControl,
                 tuneLength=10,
                 preProc = c("center", "scale")
                 )
```

第2層のknnでの予測値算出

```
knn2.model
knn2.pred=predict(knn2.model, X, type="prob")
head(knn2.pred)
write.csv(knn2.pred, "knn2.csv", row.names=F)
knn2.result=read.csv("knn2.csv", header=T)
str(knn2.result)
```

```
> knn2.pred=predict(knn2.model, X, type="prob")
> head(knn2.pred)
  Acer_Capillipes Acer_Circinatum Acer_Mono Acer_Opalus Acer_Palmatum
1               0               0         0   0.6666667             0
2               0               0         0   0.0000000             0
3               0               0         0   0.0000000             0
4               0               0         0   0.0000000             0
5               0               0         0   0.0000000             0
6               0               0         0   0.0000000             0
  Acer_Pictum Acer_Platanoids Acer_Rubrum Acer_Rufinerve Acer_Saccharinum
1           0               0           0              0                0
2           0               0           0              0                0
3           0               0           0              0                0
4           0               0           0              0                0
5           0               0           0              0                0
6           0               0           0              0                0
  Alnus_Cordata Alnus_Maximowiczii Alnus_Rubra Alnus_Sieboldiana Alnus_Viridis
1             0                  0           0                 0             0
2             0                  0           0                 0             0
3             0                  0           0                 0             0
4             0                  0           0                 0             0
5             0                  0           0                 0             0
6             0                  0           0                 0             0
  Arundinaria_Simonii Betula_Austrosinensis Betula_Pendula Callicarpa_Bodinieri
1                   0                     0              0                    0
2                   0                     0              0                    0
3                   0                     0              0                    0
```

```
> knn2.model
k-Nearest Neighbors

990 samples
299 predictors
 99 classes: 'Acer_Capillipes', 'Acer_Circinatum', 'Acer_Mono', 'Acer_Opalus',

Pre-processing: centered (299), scaled (299)
Resampling: Repeated Train/Test Splits Estimated (25 reps, 80%)
Summary of sample sizes: 792, 792, 792, 792, 792, 792, ...
Resampling results across tuning parameters:

  k   Accuracy   Kappa
   5  1.0000000  1.0000000
   7  1.0000000  1.0000000
   9  1.0000000  1.0000000
  11  1.0000000  1.0000000
  13  1.0000000  1.0000000
  15  1.0000000  1.0000000
  17  0.9945455  0.9944898
  19  0.9921212  0.9920408
  21  0.9901010  0.9900000
  23  0.9878788  0.9877551

Accuracy was used to select the optimal model using  the largest value.
The final value used for the model was k = 15.
```

モデルを保存する

#モデルを保存する(こうするとカレントディレクトリにrf2というファイルができる)

saveRDS(knn2.model, file="knn2")

#読み込むときは以下のようにして読み込む

model.read <- readRDS(file="knn2")

```
> model.read
k-Nearest Neighbors

990 samples
299 predictors
 99 classes: 'Acer_Capillipes', 'Acer_Circinatum', 'Acer_Mono', 'Acer_Opalus',

Pre-processing: centered (299), scaled (299)
Resampling: Repeated Train/Test Splits Estimated (25 reps, 80%)
Summary of sample sizes: 792, 792, 792, 792, 792, 792, ...
Resampling results across tuning parameters:

  k   Accuracy   Kappa
   5  1.0000000  1.0000000
   7  1.0000000  1.0000000
   9  1.0000000  1.0000000
  11  1.0000000  1.0000000
  13  1.0000000  1.0000000
  15  1.0000000  1.0000000
  17  0.9945455  0.9944898
  19  0.9921212  0.9920408
  21  0.9901010  0.9900000
  23  0.9878788  0.9877551

Accuracy was used to select the optimal model using  the largest value.
The final value used for the model was k = 15.
```

```
> write.csv(knn2.pred, "knn2.csv", row.names=F)
> head(read.csv("knn2.csv",header=T))
  Acer_Capillipes Acer_Circinatum Acer_Mono Acer_Opalus Acer_Palmatum
1               0               0         0   0.6666667             0
2               0               0         0   0.0000000             0
3               0               0         0   0.0000000             0
4               0               0         0   0.0000000             0
5               0               0         0   0.0000000             0
6               0               0         0   0.0000000             0
  Acer_Pictum Acer_Platanoids Acer_Rubrum Acer_Rufinerve Acer_Saccharinum
1           0               0           0              0                0
2           0               0           0              0                0
3           0               0           0              0                0
4           0               0           0              0                0
5           0               0           0              0                0
6           0               0           0              0                0
  Alnus_Cordata Alnus_Maximowiczii Alnus_Rubra Alnus_Sieboldiana Alnus_Viridis
1             0                  0           0                 0             0
2             0                  0           0                 0             0
3             0                  0           0                 0             0
4             0                  0           0                 0             0
5             0                  0           0                 0             0
6             0                  0           0                 0             0
  Arundinaria_Simonii Betula_Austrosinensis Betula_Pendula Callicarpa_Bodinieri
1                   0                     0              0                    0
2                   0                     0              0                    0
3                   0                     0              0                    0
```

第2層のxgboostでのモデル作成

学習用データの目的変数をxgboost用に加工

```
# 目的変数
y <- Y
#Factor型である目的変数を、0,1,2,3,,,98, の整数にする。
#xgboost で既定されいるクラスは 0 baseなため。
y <- as.integer(y)-1
```

学習用データの説明変数をxgboost用に加工

```
# xgboost を使うときのため行列に変換
x <- as.matrix(X)
```

パラメータ指定

```
set.seed(123) # 固定シードで試す
param <- list("objective" = "multi:softprob",
              # 多クラスの分類で各クラスに所属する確率を求める
              "eval_metric" = "mlogloss", # 損失関数の設定
               "num_class" = 99 # class が今回は99個存在。
              )
```

予測をする前に最適な木の数を探す

#スタージェスの公式

k<-round(1+log2(nrow(X)))

#適当な数を入れる

cv.nround <- 50 #search

#nroundsを決めるためのモデル作成

bst.cv <- xgb.cv(param=param, data = x, label = y, nfold = k, nrounds=cv.nround)

```
[13]    train-mlogloss:0.082024+0.000330        test-mlogloss:0.095687+0.004944
[14]    train-mlogloss:0.068907+0.000280        test-mlogloss:0.080558+0.004261
[15]    train-mlogloss:0.058593+0.000240        test-mlogloss:0.068628+0.003701
[16]    train-mlogloss:0.053505+0.000417        test-mlogloss:0.064559+0.003462
[17]    train-mlogloss:0.053504+0.000417        test-mlogloss:0.064589+0.003463
[18]    train-mlogloss:0.053504+0.000417        test-mlogloss:0.064616+0.003464
[19]    train-mlogloss:0.053504+0.000417        test-mlogloss:0.064640+0.003465
[20]    train-mlogloss:0.053503+0.000417        test-mlogloss:0.064663+0.003465
[21]    train-mlogloss:0.053503+0.000417        test-mlogloss:0.064683+0.003466
[22]    train-mlogloss:0.053503+0.000417        test-mlogloss:0.064701+0.003467
[23]    train-mlogloss:0.053503+0.000417        test-mlogloss:0.064718+0.003467
[24]    train-mlogloss:0.053502+0.000417        test-mlogloss:0.064733+0.003467
[25]    train-mlogloss:0.053502+0.000417        test-mlogloss:0.064746+0.003468
[26]    train-mlogloss:0.053502+0.000417        test-mlogloss:0.064758+0.003468
[27]    train-mlogloss:0.053502+0.000417        test-mlogloss:0.064769+0.003469
[28]    train-mlogloss:0.053502+0.000417        test-mlogloss:0.064780+0.003469
[29]    train-mlogloss:0.053502+0.000417        test-mlogloss:0.064789+0.003469
[30]    train-mlogloss:0.053502+0.000417        test-mlogloss:0.064797+0.003470
[31]    train-mlogloss:0.053502+0.000417        test-mlogloss:0.064804+0.003470
[32]    train-mlogloss:0.053502+0.000417        test-mlogloss:0.064811+0.003470
[33]    train-mlogloss:0.053502+0.000417        test-mlogloss:0.064817+0.003470
[34]    train-mlogloss:0.053502+0.000417        test-mlogloss:0.064822+0.003470
[35]    train-mlogloss:0.053502+0.000417        test-mlogloss:0.064827+0.003471
[36]    train-mlogloss:0.053502+0.000417        test-mlogloss:0.064832+0.003471
[37]    train-mlogloss:0.053502+0.000417        test-mlogloss:0.064836+0.003471
[38]    train-mlogloss:0.053502+0.000417        test-mlogloss:0.064840+0.003471
[39]    train-mlogloss:0.053502+0.000417        test-mlogloss:0.064843+0.003471
```

```
set.seed(123)
nround <- 17
# モデルの構築
bst2 <- xgboost(param=param, data = x, label = y, nrounds=nround)
```

モデルを保存する

#モデルを保存する(こうするとカレントディレクトリにbst2というファイルができる)

saveRDS(bst2, file="bst2")

#読み込むときは以下のようにして読み込む

model.read <- readRDS(file="bst2")

test=as.matrix(X)

pred <- predict(bst2,test)

head(pred)

str(pred)

```
> test=as.matrix(test)
> pred <- predict(bst,test[,-1])
> head(pred)
[1] 0.008701433 0.003948880 0.002994578 0.042810339 0.004569929 0.004519264
> str(pred)
 num [1:58806] 0.0087 0.00395 0.00299 0.04281 0.00457 ...
```

一列のベクトル。。。このままでは、Kaggleに提出できない。。

#99列の行列にする。数字の配列は左から右

```
q=matrix(pred,ncol=99, byrow=T)
str(q)
```

new=data.frame(q)　#データフレーム型にする。

列名を付ける。

#列順は、きっとABC順(sample_submission.csvも同じ)
#訓練データの植物の名前をとってくる。
y <- Y
#重複をunique関数で取り除き、Factor型を整数型にする。
z=as.integer(unique(y))

列名を付ける。

#植物名とFactorとしての数字をペアにする(数字が実体で名前が植物名) 。

names(z)=unique(y)

列名を付ける。

#数字でソートする(昇順)

z1=sort(z)

列名を付ける。

```
#列に名前を付ける。
colnames(new)=names(z1)
head(new)
```

今のデータをcsvファイルとして出力

```
write.csv(new,"xgboost2.csv", row.names=F)
```

第3層のための説明変数作成

```
xgboost2.result=read.csv("xgboost2.csv", header=T)
str(xgboost2.result)
knn2.result=read.csv("knn2.csv", header=T)
str(knn2.result)
new2.X=cbind(xgboost2.result,knn2.result)
str(new2.X)
```

第3層のための説明変数作成

tsne1.result=read.csv("tsne1.csv", header=T)

str(tsne1.result)

new2.X=cbind(new2.X,tsne1.result)

dim(new2.X)

X=new2.X

```
> new2.X=cbind(new2.X,tsne1.result)
> dim(new2.X)
[1] 990 200
> X=new2.X
```

```
> xgboost2.result=read.csv("xgboost2.csv", header=T)
> str(xgboost2.result)
'data.frame':   990 obs. of  99 variables:
 $ Acer_Capillipes              : num  0.00051 0.000409 0.001803 0.000435 0.0004
 $ Acer_Circinatum              : num  0.000456 0.000366 0.000364 0.000414 0.000
 $ Acer_Mono                    : num  0.000458 0.000367 0.000365 0.000416 0.000
 $ Acer_Opalus                  : num  0.953238 0.000396 0.000393 0.000448 0.000
 $ Acer_Palmatum                : num  0.000458 0.000367 0.000365 0.000416 0.000
 $ Acer_Pictum                  : num  0.000457 0.000366 0.000364 0.000415 0.000
 $ Acer_Platanoids              : num  0.00046 0.000369 0.000367 0.000418 0.0003
 $ Acer_Rubrum                  : num  0.000458 0.000368 0.000365 0.000416 0.000
 $ Acer_Rufinerve               : num  0.000458 0.000368 0.000365 0.000416 0.000
 $ Acer_Saccharinum             : num  0.000457 0.000367 0.000365 0.000416 0.000
 $ Alnus_Cordata                : num  0.000458 0.001737 0.000366 0.000416 0.000
 $ Alnus_Maximowiczii           : num  0.000456 0.000366 0.000364 0.000414 0.000
 $ Alnus_Rubra                  : num  0.000456 0.000366 0.000364 0.000414 0.000
 $ Alnus_Sieboldiana            : num  0.000454 0.000364 0.000362 0.000413 0.000
 $ Alnus_Viridis                : num  0.000456 0.000366 0.000364 0.000414 0.000
```

```
> knn2.result=read.csv("knn2.csv", header=T)
> str(knn2.result)
'data.frame':   990 obs. of  99 variables:
 $ Acer_Capillipes              : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Acer_Circinatum              : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Acer_Mono                    : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Acer_Opalus                  : num  0.667 0 0 0 0 ...
 $ Acer_Palmatum                : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Acer_Pictum                  : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Acer_Platanoids              : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Acer_Rubrum                  : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Acer_Rufinerve               : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Acer_Saccharinum             : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Alnus_Cordata                : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Alnus_Maximowiczii           : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Alnus_Rubra                  : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Alnus_Sieboldiana            : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Alnus_Viridis                : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Arundinaria_Simonii          : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Betula_Austrosinensis        : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Betula_Pendula               : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Callicarpa_Bodinieri         : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Castanea_Sativa              : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Celtis_Koraiensis            : num  0 0 0 0 0 0 0 0 0 0 ...
```

```
> new2.X=cbind(xgboost2.result,knn2.result)
> str(new2.X)
'data.frame':   990 obs. of  198 variables:
 $ Acer_Capillipes              : num  0.00051 0.000409 0.001803 0.000435 0.0004$
 $ Acer_Circinatum              : num  0.000456 0.000366 0.000364 0.000414 0.000$
 $ Acer_Mono                    : num  0.000458 0.000367 0.000365 0.000416 0.000$
 $ Acer_Opalus                  : num  0.953238 0.000396 0.000393 0.000448 0.000$
 $ Acer_Palmatum                : num  0.000458 0.000367 0.000365 0.000416 0.000$
 $ Acer_Pictum                  : num  0.000457 0.000366 0.000364 0.000415 0.000$
 $ Acer_Platanoids              : num  0.00046 0.000369 0.000367 0.000418 0.0003$
 $ Acer_Rubrum                  : num  0.000458 0.000368 0.000365 0.000416 0.000$
 $ Acer_Rufinerve               : num  0.000458 0.000368 0.000365 0.000416 0.000$
 $ Acer_Saccharinum             : num  0.000457 0.000367 0.000365 0.000416 0.000$
 $ Alnus_Cordata                : num  0.000458 0.001737 0.000366 0.000416 0.000$
 $ Alnus_Maximowiczii           : num  0.000456 0.000366 0.000364 0.000414 0.000$
 $ Alnus_Rubra                  : num  0.000456 0.000366 0.000364 0.000414 0.000$
 $ Alnus_Sieboldiana            : num  0.000454 0.000364 0.000362 0.000413 0.000$
 $ Alnus_Viridis                : num  0.000456 0.000366 0.000364 0.000414 0.000$
 $ Arundinaria_Simonii          : num  0.000478 0.000384 0.000382 0.000435 0.000$
 $ Betula_Austrosinensis        : num  0.000457 0.000367 0.000365 0.000415 0.001$
 $ Betula_Pendula               : num  0.000455 0.000365 0.000363 0.000414 0.000$
 $ Callicarpa_Bodinieri         : num  0.000456 0.000366 0.000364 0.000414 0.000$
 $ Castanea_Sativa              : num  0.000455 0.000365 0.000363 0.000414 0.000$
 $ Celtis_Koraiensis            : num  0.000458 0.000367 0.001724 0.000416 0.000$
 $ Cercis_Siliquastrum          : num  0.000516 0.000378 0.000412 0.000428 0.000$
```

```
> tsne1.result=read.csv("tsne1.csv", header=T)
> str(tsne1.result)
'data.frame':    990 obs. of  2 variables:
 $ V1: num  -11.45 22.87 4.15 -1.41 -58.17 ...
 $ V2: num  14.95 17.64 -31.43 -16.63 -4.16 ...
```

第3層でのxgboostによるモデル作成

学習用データの目的変数をxgboost用に加工

```
# 目的変数
y <- Y
#Factor型である目的変数を、0,1,2,3,,,98, の整数にする。
#xgboost で既定されいるクラスは 0 baseなため。
y <- as.integer(y)-1
```

学習用データの説明変数をxgboost用に加工

```
# xgboost を使うときのため行列に変換
x <- as.matrix(X)
```

パラメータ指定

```
set.seed(123) # 固定シードで試す
param <- list("objective" = "multi:softprob",
              # 多クラスの分類で各クラスに所属する確率を求める
              "eval_metric" = "mlogloss", # 損失関数の設定
               "num_class" = 99 # class が今回は99個存在。
              )
```

予測をする前に最適な木の数を探す

```
#スタージェスの公式
k<-round(1+log2(nrow(X)))
#適当な数を入れる
cv.nround <- 50 #search
#nroundsを決めるためのモデル作成
bst.cv <- xgb.cv(param=param, data = x, label = y,  nfold = k,
nrounds=cv.nround)
```

```
[8]     train-mlogloss:0.247748+0.000964        test-mlogloss:0.312527+0.017810
[9]     train-mlogloss:0.195244+0.000749        test-mlogloss:0.248523+0.014704
[10]    train-mlogloss:0.155675+0.000589        test-mlogloss:0.199727+0.012184
[11]    train-mlogloss:0.125644+0.000470        test-mlogloss:0.162299+0.010148
[12]    train-mlogloss:0.102684+0.000380        test-mlogloss:0.133413+0.008508
[13]    train-mlogloss:0.084990+0.000312        test-mlogloss:0.110967+0.007189
[14]    train-mlogloss:0.071233+0.000260        test-mlogloss:0.093391+0.006126
[15]    train-mlogloss:0.060435+0.000220        test-mlogloss:0.079509+0.005268
[16]    train-mlogloss:0.053242+0.000420        test-mlogloss:0.071180+0.004573
[17]    train-mlogloss:0.053240+0.000419        test-mlogloss:0.071198+0.004571
[18]    train-mlogloss:0.053239+0.000419        test-mlogloss:0.071214+0.004569
[19]    train-mlogloss:0.053238+0.000419        test-mlogloss:0.071229+0.004567
[20]    train-mlogloss:0.053237+0.000419        test-mlogloss:0.071242+0.004566
[21]    train-mlogloss:0.053236+0.000419        test-mlogloss:0.071254+0.004565
[22]    train-mlogloss:0.053235+0.000419        test-mlogloss:0.071266+0.004563
[23]    train-mlogloss:0.053235+0.000419        test-mlogloss:0.071276+0.004562
[24]    train-mlogloss:0.053235+0.000419        test-mlogloss:0.071286+0.004562
[25]    train-mlogloss:0.053234+0.000419        test-mlogloss:0.071294+0.004561
[26]    train-mlogloss:0.053234+0.000419        test-mlogloss:0.071302+0.004560
[27]    train-mlogloss:0.053233+0.000419        test-mlogloss:0.071309+0.004560
[28]    train-mlogloss:0.053233+0.000419        test-mlogloss:0.071315+0.004559
[29]    train-mlogloss:0.053233+0.000419        test-mlogloss:0.071321+0.004559
[30]    train-mlogloss:0.053233+0.000419        test-mlogloss:0.071326+0.004558
[31]    train-mlogloss:0.053233+0.000419        test-mlogloss:0.071331+0.004558
[32]    train-mlogloss:0.053233+0.000419        test-mlogloss:0.071335+0.004558
```

```
set.seed(123)
nround <- 17
# モデルの構築
bst3 <- xgboost(param=param, data = x, label = y, nrounds=nround)
```

モデルを保存する

#モデルを保存する(こうするとカレントディレクトリにbst3というファイルができる)

saveRDS(bst3, file="bst3")

#読み込むときは以下のようにして読み込む

model.read <- readRDS(file="bst3")

ここでテストデータを用いる

- テストデータ読み込み
- 第1層での予測値の算出
- Tsneでの特徴量の算出
- 第2層のための説明変数の作成
- 第2層での予測値算出
- 第3層で使うための説明変数の作成
- 第3層でのrandomforestによる予測→kaggleへ提出
- 第3層でのxgboostによる予測→kaggleへ提出

テスト用データ

```
data2=read.csv("test.csv", header=T)
dim(data2)
str(data2)
test.id=data2[,1]
test.X=data2[,2:193]
```

```
> data2=read.csv("test.csv", header=T)
> dim(data2)
[1] 594 193
> str(data2)
'data.frame':   594 obs. of  193 variables:
 $ id       : int  4 7 9 12 13 16 19 23 24 28 ...
 $ margin1  : num  0.01953 0.00781 0 0 0.00195 ...
 $ margin2  : num  0.00977 0.00586 0 0 0 ...
 $ margin3  : num  0.07812 0.06445 0.00195 0.00977 0.01562 ...
 $ margin4  : num  0.01172 0.00977 0.02148 0.01172 0.00977 ...
 $ margin5  : num  0.00391 0.00391 0.04102 0.01758 0.03906 ...
 $ margin6  : num  0.0156 0.0137 0 0 0 ...
 $ margin7  : num  0.00586 0.00781 0.02344 0.00391 0.00977 ...
 $ margin8  : num  0 0 0 0 0 0 0 0 0 0 ...
 $ margin9  : num  0.00586 0.0332 0.01172 0.00391 0.00586 ...
 $ margin10 : num  0.02344 0.02344 0.00586 0.00195 0 ...
 $ margin11 : num  0.00586 0.00977 0.00195 0 0.00195 ...
 $ margin12 : num  0.0215 0.0195 0.0215 0.0293 0.0332 ...
 $ margin13 : num  0.07617 0.03906 0.00195 0 0 ...
 $ margin14 : num  0.00195 0.02734 0.01953 0.03906 0.00391 ...
 $ margin15 : num  0.0352 0.0176 0.0352 0.0371 0.0117 ...
 $ margin16 : num  0 0 0 0.00391 0 ...
 $ margin17 : num  0.00195 0.01758 0.00781 0.00781 0.00391 ...
 $ margin18 : num  0.02148 0.01562 0.00195 0.00586 0.00586 ...
 $ margin19 : num  0.00195 0.00977 0.04688 0.00781 0.01172 ...
```

第1層目のテストデータによる予測値算出

```
library(caret)
rf.model <- readRDS(file="rf1")
rf.pred=predict(rf.model,test.X,type="prob")
write.csv(rf.pred, "predrf.csv",row.names=F )
knn.model <- readRDS(file="knn1")
knn.pred=predict(knn.model,test.X,type="prob")
write.csv(knn.pred, "predknn.csv",row.names=F )
```

第一層目のテストデータによる予測値算出

```
test=as.matrix(test.X)
bst<- readRDS(file="bst1")
pred <- predict(bst,test)
head(pred)
str(pred)
```

#99列の行列にする。数字の配列は左から右

```
q=matrix(pred,ncol=99, byrow=T)
str(q)
```

new=data.frame(q)　#データフレーム型にする。

列名を付ける。

#列順は、きっとABC順(sample_submission.csvも同じ)
#訓練データの植物の名前をとってくる。
y <- Y
#重複をunique関数で取り除き、Factor型を整数型にする。
z=as.integer(unique(y))

列名を付ける。

#植物名とFactorとしての数字をペアにする(数字が実体で名前が植物名) 。

names(z)=unique(y)

列名を付ける。

#数字でソートする(昇順)

z1=sort(z)

列名を付ける。

```
#列に名前を付ける。
colnames(new)=names(z1)
head(new)
```

```
> colnames(new)=names(z1)
> head(new)
  Acer_Capillipes Acer_Circinatum    Acer_Mono  Acer_Opalus Acer_Palmatum
1    0.0087014334     0.003948880 0.0029945783 0.0428103395  0.0045699291
2    0.0013931614     0.001527575 0.0014395113 0.0019358563  0.0044749202
3    0.0015701300     0.697212934 0.0035078584 0.0018159703  0.0031361626
4    0.0004623707     0.002009790 0.0002976838 0.0006155035  0.0009532305
5    0.0008586111     0.001793819 0.0003878254 0.0006186561  0.0005918475
6    0.0069629666     0.006093140 0.0046206494 0.1992146969  0.0070514213
   Acer_Pictum Acer_Platanoids  Acer_Rubrum Acer_Rufinerve Acer_Saccharinum
1 0.0045192642    0.0045692055 0.0035081625   0.0037453352     0.0041058604
2 0.0014665206    0.2983400524 0.0013570888   0.0012959896     0.0059904354
3 0.0014145926    0.0017039666 0.0026222945   0.0045404974     0.0016497591
4 0.0003353161    0.0005179175 0.0006124873   0.0006640864     0.0050293286
5 0.0004368530    0.0044814851 0.0006623237   0.0160320010     0.0006532824
6 0.0052047777    0.0062507531 0.0054131122   0.0051694009     0.0063353614
  Alnus_Cordata Alnus_Maximowiczii  Alnus_Rubra Alnus_Sieboldiana Alnus_Viridis
1  0.0047903359       0.0035092884 0.0050556096      0.0037318920   0.003765247
2  0.0013486104       0.0013575243 0.0042403811      0.0014436357   0.001456539
3  0.0012442512       0.0016108741 0.0024747071      0.0012205307   0.005643236
4  0.0003465592       0.0006480622 0.0205081217      0.0039502676   0.000776135
5  0.0005615338       0.0285057500 0.0009154832      0.0006009013   0.872102737
```

今のデータをcsvファイルとして出力

```
write.csv(new,"predxgboost1.csv", row.names=F)
```

T-SNE特徴量の算出

```
library(tsne)
tsne.X = tsne(test.X)
tsne.X
write.csv(tsne.X, "tsne2.csv",row.names=F)
```

```
> library(tsne)
> tsne.X = tsne(test.X)
sigma summary: Min. : 0.51 |1st Qu. : 0.7042 |Median : 0.7701 |Mean : 0.7783 |3$
Epoch: Iteration #100 error is: 16.58765187092
Epoch: Iteration #200 error is: 0.64706044311565
Epoch: Iteration #300 error is: 0.552174405813906
Epoch: Iteration #400 error is: 0.532732510855766
Epoch: Iteration #500 error is: 0.527660358256184
Epoch: Iteration #600 error is: 0.525649241184139
Epoch: Iteration #700 error is: 0.524513586822056
Epoch: Iteration #800 error is: 0.523573601044074
Epoch: Iteration #900 error is: 0.522759312469273
Epoch: Iteration #1000 error is: 0.522137293031794
```

```
> tsne.X
                [,1]          [,2]
 [1,] -17.79611326  18.24887856
 [2,]   8.55064542   6.58745527
 [3,]   7.95229408 -27.32813575
 [4,]  19.65228826  -4.66605661
 [5,]  18.52051932 -23.92253431
 [6,]   8.44040535  22.93594799
 [7,]   6.24838118  19.01139764
 [8,] -27.27881029  15.93771555
 [9,] -32.43831597 -37.01622365
[10,]  -0.76013132 -15.80005332
[11,]  22.74679864  17.31560646
[12,] -11.69916163 -18.30843374
[13,]   2.62212288  37.83281556
[14,]  -8.58709610  34.67029332
[15,]  27.30709287   9.19685248
[16,]  14.60916223 -13.19201300
[17,] -49.07600392   7.63201168
[18,]  14.70362954  45.44399116
[19,]  20.68908364   0.25527463
[20,]   4.78146389 -54.20473243
[21,] -49.10368736  24.57655877
```

2層目の学習器によるテストデータを用いた予測値の作り方

- 説明変数を作る。
- Randomforestによる学習
- Knnによる学習
- Xgboostによる学習

説明変数を作る

```
rf.result=read.csv("predrf.csv", header=T)
str(rf.result)
new.X=rf.result
str(new.X)
```

説明変数を作る

```
knn.result=read.csv("predknn.csv", header=T)
str(knn.result)
new.X=cbind(new.X,knn.result)
dim(new.X)
```

説明変数を作る

```
xgboost.result=read.csv("predxgboost1.csv", header=T)
str(xgboost.result)
new.X=cbind(new.X, xgboost.result)
dim(new.X)
tsne2.result=read.csv("tsne2.csv", header=T)
str(tsne2.result)
new.X=cbind(new.X, tsne2.result)
X=new.X
dim(X)
```

```
> X=new.X
> dim(X)
[1] 594 299
```

```
write.csv(X,"explain2.csv",row.names=F)
```

説明変数を第2層目のモデルに当てはめる

- Knn(モデル名 : knn2.model)
- Xgboost(モデル名 : bst2)

第2層目のknnモデルへの当てはめ

knn2.model<- readRDS(file="knn2")

knn.pred2=predict(knn2.model, X, type="prob")

write.csv(knn.pred2, "predknn2.csv",row.names=F)

```
> knn2.model<- readRDS(file="knn2")
> knn.pred2=predict(knn2.model, X, type="prob")
Error in `[.data.frame`(newdata, , object$method$center, drop = FALSE) :
  undefined columns selected
```

第2層目のxgboostモデルへの当てはめ

```
test=as.matrix(X)
pred <- predict(bst2,test)
q=matrix(pred,ncol=99, byrow=T)
new=data.frame(q)  #データフレーム型にする。
y <- Y
z=as.integer(unique(y))
names(z)=unique(y)
z1=sort(z)
colnames(new)=names(z1)
write.csv(new,"predxgboost2.csv", row.names=F)
```

ここまでで3つの予測ファイルができた。

predrf2.csv

predknn2.csv

predxgboost2.csv

これに加え、t-SNEの特徴量を使用する。

tsne2.csv

第3層の説明変数の作成

```
rf.result3=read.csv("predrf2.csv", header=T)
str(rf.result3)
knn.result3=read.csv("predknn2.csv", header=T)
str(knn.result3)
new.X=cbind(rf.result3,knn.result3)
dim(new.X)
```

第3層の説明変数の作成

```
xgboost.result3=read.csv("predxgboost2.csv", header=T)
str(xgboost.result3)
new.X=cbind(new.X,xgboost.result3)
dim(new.X)
tsne.result3=read.csv("tsne2.csv", header=T)
str(tsne.result3)
new.X=cbind(new.X,tsne.result3)
dim(new.X)
X=new.X
```

第3層のrandomForestでのtestデータを用いた予測。

```
rfpredfinal=predict(rf3.model,X,type="prob")
dim(rfpredfinal)
head(rfpredfinal)
row.name(rfpredfinal)=stest.id
write.csv(rfpredfinal, "ensemblerf1.csv",row.names=F)
#もしくは
write.csv(rfpredfinal, "ensemblerf2.csv",row.names=T)
#この後に、ファイルを開いて、id名を付ける。
```

第3層目のxgboostモデルへの当てはめ

```
test=as.matrix(X)
pred <- predict(bst3,test)
q=matrix(pred,ncol=99, byrow=T)
new=data.frame(q)  #データフレーム型にする。
rownames(q)=test.id
y <- Y
z=as.integer(unique(y))
names(z)=unique(y)
z1=sort(z)
colnames(new)=names(z1)
write.csv(new,"emblexgboost1.csv", row.names=F)
#この後で、ファイルを開いて、idを入れる。
```

終わり。