STAT482_final

Sang Chan Lee

4/25/2020

```
## Loading required package: Matrix

## Loaded glmnet 3.0-2

## Loading required package: lattice

## Loading required package: ggplot2

## Loading required package: nlme

## Loading required package: nnet

## Loading required package: survival

##
## Attaching package: 'survival'

## The following object is masked from 'package:caret':
##
##     cluster

## Package 'mclust' version 5.4.5
## Type 'citation("mclust")' for citing this R package in publications.

## Warning: package 'factoextra' was built under R version 3.6.2

## Welcome! Want to learn more? See two factoextra-related books at
https://goo.gl/ve3WBa
```

Setup and data import

data description

There are total 61 covariates and 850 counties from Swing States, which are Colorado, Florida, Iowa, Michigan, Minnesota, Nevada, New Hampshire, North Carolina, Ohio, Pennsylvania, Virginia, and Wisconsin. For variable selection, Lasso regression is conducted with largest value of lambda such that error is within 1 standard error of the minimum. As a result, 11 variables are selected. Log odds ratio, $log(\frac{p(x)}{1-p(x)})$ for the response variable is used for non-parametric regression and 11 covariates are standardized.

Lasso Regression

Lasso regression is a type of linear regression that uses shrinkage using L1 Regularization. This particular type of regression is well-suited for models showing high levels of muticollinearity or when you want to automate certain parts of model selection, like variable selection/parameter elimination. The goal of the lasso algorithm is to minimize the quantity,

$$(\sum_{i=1}^{n} y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij})^2 + \lambda \sum_{j=1}^{p} = |\beta_j| = RSS + \lambda||\beta||_1$$

Partitioning data

Data is partitioned into train(75%) and test(25%) data sets.

```
set.seed(101)
indexing <- createDataPartition(y = data$y, p = 0.75,list = FALSE)
data_train = data[indexing,]
data_test = data[-indexing,]
```

K-nearest neighbor

The tuning parameter k=23 is used for K-nearest neighbor by Cross Validation.

```
knn_fit = train(y~.,
                data = data_train,
                trControl = trainControl(method = 'repeatedcv', number = 10,
repeats = 3),
                method = 'knn',
                tuneLength = 10
                )
knn_pred = predict(knn_fit, newdata = data_test)
knn_RMSE = sqrt(mean((data_test$y - knn_pred)^2))
```

Random Forest

The tuning parameters, the number of trees and number of variables randomly sampled as candiates at each split are 500 and 6 respectively by Cross Validation.

```
rf_fit = train(y~.,
               data = data_train,
               trControl = trainControl(method = 'repeatedcv', number = 10,
repeats = 3),
               method = 'rf',
               tuneLength = 5
```

```
                )
rf_pred = predict(rf_fit, newdata = data_test)
rf_RMSE = sqrt(mean((data_test$y - rf_pred)^2))
```

Gradient Boost

The tuning parameters, the number of trees is 50, the maximum depth of each tree is 5, and shrinkage is 0.1 are used by Cross Validation.

```
boost.fit <- train(y ~ .,
                   data = data_train,
                   method = "gbm",
                   trControl = trainControl(method = 'repeatedcv', number =
10, repeats = 3),
                   verbose = FALSE,
                   tuneLength = 10
                   )
boost_pred = predict(boost.fit, newdata = data_test)
boost_RMSE = sqrt(mean((data_test$y - boost_pred)^2))
```

Baysian Additive Regression Trees

```
bart_fit = gbart(x.train = data_train[,-1], y.train = data_train$y, type =
'wbart')
```

```
## *****Calling gbart: type=1
## *****Data:
## data:n,p,np: 638, 11, 0
## y1,yn: 0.113635, -0.024503
## x1,x[n*p]: 1.470566, -0.242065
## *****Number of Trees: 200
## *****Number of Cut Points: 100 ... 100
## *****burn,nd,thin: 100,1000,1
## *****Prior:beta,alpha,tau,nu,lambda,offset:
2,0.95,0.0208953,3,0.00625992,-0.389933
## *****sigma: 0.179267
## *****w (weights): 1.000000 ... 1.000000
## *****Dirichlet:sparse,theta,omega,a,b,rho,augment: 0,0,1,0.5,1,11,0
## *****printevery: 100
##
## MCMC
## done 0 (out of 1100)
## done 100 (out of 1100)
## done 200 (out of 1100)
## done 300 (out of 1100)
## done 400 (out of 1100)
## done 500 (out of 1100)
## done 600 (out of 1100)
## done 700 (out of 1100)
```

```
## done 800 (out of 1100)
## done 900 (out of 1100)
## done 1000 (out of 1100)
## time: 4s
## trcnt,tecnt: 1000,0

bart_pred = predict(bart_fit, newdata = data_test[,-1])

## *****In main of C++ for bart prediction
## tc (threadcount): 1
## number of bart draws: 1000
## number of trees in bart sum: 200
## number of x columns: 11
## from x,np,p: 11, 212
## ***using serial code

pred.mean=apply(bart_pred,2,mean)
bart_RMSE = sqrt(mean((data_test$y - pred.mean)^2))
```

RMSE from knn, random forest, gradient boost and bart

Among four non-parametric regression models, Gradient Boost has the lowest RMSE. Thus it is used for the prediction.

```
RMSE = cbind(knn_RMSE, rf_RMSE, boost_RMSE, bart_RMSE)
RMSE

##       knn_RMSE   rf_RMSE boost_RMSE bart_RMSE
## [1,] 0.1813547 0.1567539  0.1560803 0.1582998
```

Data preprocessiong for prediction

```
data2018 = pred2018[,c(-1,-2,-3)]
data2014 = pred2014[,c(-1,-2,-3)]

for (i in 1:ncol(data2018)){
  data2018[,i] = as.numeric(data2018[,i])
}


data2018 = apply(data2018, 2, scale)
data2014 = apply(data2014, 2, scale)

data_pred = data2018 - data2014
colnames(data_pred) = colnames(data)[-1]
```

2020 prediction

```
pred = predict(boost.fit, newdata = data_pred)
logOdds_dem2018 = pred + logOdds_dem2014

prob2018 = exp(logOdds_dem2018) / (1+exp(logOdds_dem2018))

aa = c()
for (i in 1:length(prob2018)){
  if (prob2018[i] > 0.5){
    aa[i] = 0
  }
  else
    aa[i] = 1
}

# 0 for dem and 1 for gop
vote_pred = data.frame(County = pred2018[,2], State = pred2018[,3], pred =
aa)
vote_pred[1:10,]
```

```
##               County     State pred
## 1       Adams County  Colorado    1
## 2     Alamosa County  Colorado    1
## 3    Arapahoe County  Colorado    1
## 4   Archuleta County  Colorado    1
## 5        Baca County  Colorado    1
## 6        Bent County  Colorado    1
## 7     Boulder County  Colorado    0
## 8  Broomfield County  Colorado    1
## 9     Chaffee County  Colorado    1
## 10   Cheyenne County  Colorado    1
```

Principal Component Analysis

For reduction of dimensionality of data, Principal Component Analysis(PCA) is conducted and 6 principal components are used so that 97.49% of variation is explained.

```
pca_out = prcomp(data2018)
summary(pca_out)$importance[3,]
```

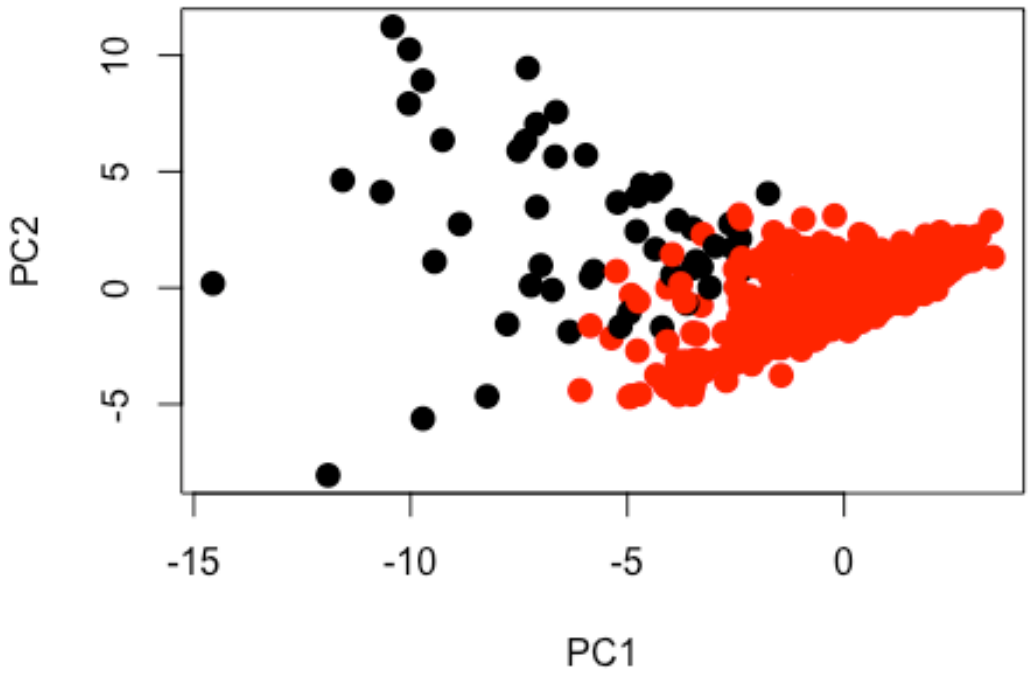
```
##     PC1     PC2     PC3     PC4     PC5     PC6     PC7     PC8     PC9
PC10
## 0.47419 0.72258 0.81017 0.88393 0.94120 0.96233 0.97340 0.98257 0.99054
0.99687
##    PC11
## 1.00000
```

```
data_cluster = pca_out$x[,1:6]
```

K-mean clustering

By Siluhouette method, it is suggested to use two clusters for K-mean clustering model. In comparison to prediction from Gradient Boost, 784 counties have the same result.

```
kmean_fit <- kmeans(data, centers = 2, nstart = 10)
plot(data_cluster,col=kmean_fit$cluster,cex=1.5,pch=16,lwd=2)
```




```
reg_kmean = data.frame(aa, bb = kmean_fit$cluster-1)
length(which(aa == kmean_fit$cluster-1))

## [1] 784
```

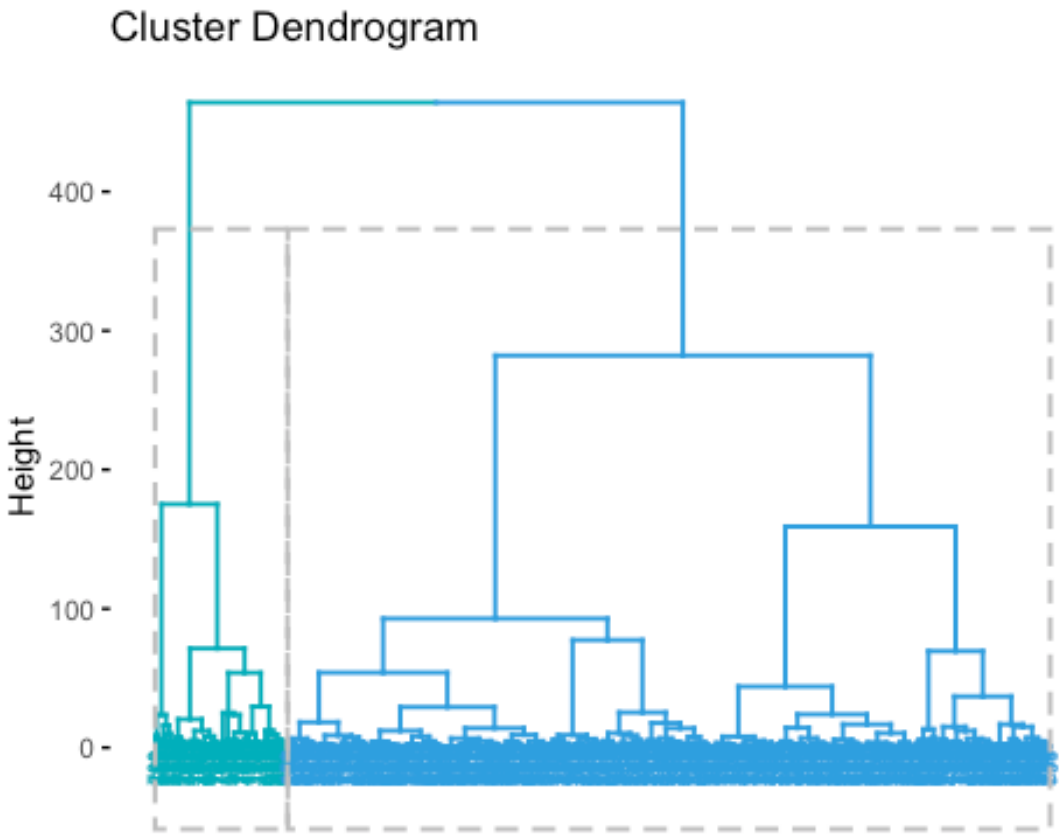
Hierarchical clustering

By Siluhouette method, it is suggested to use two clusters for hierarchical clustering model. In comparison to prediction from Gradient Boost, 735 counties have the same result.

```
hclust_ward = hcut(data_cluster, k = 2, hc_method = 'ward.D')

plot_ward = fviz_dend(hclust_ward, rect = TRUE, cex = 0.5,
```

```
          k_colors = c("#00AFBB","#2E9FDF"))
plot_ward
```




```
red_h = data.frame(aa, bb = hclust_ward$cluster -1)
length(which(aa == hclust_ward$cluster -1))

## [1] 735
```