Analytics of County election data year 2016 and year 2020 using various models

Group\_7

29 April 2021

library(ggplot2)  
library(tidyverse)

## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v tibble 3.0.5 v dplyr 1.0.3  
## v tidyr 1.1.2 v stringr 1.4.0  
## v readr 1.4.0 v forcats 0.5.1  
## v purrr 0.3.4

## Warning: package 'stringr' was built under R version 4.0.4

## -- Conflicts ------------------------------------------ tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

library(glmnet)

## Warning: package 'glmnet' was built under R version 4.0.4

## Loading required package: Matrix

##   
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':  
##   
## expand, pack, unpack

## Loaded glmnet 4.1-1

library(pls)

## Warning: package 'pls' was built under R version 4.0.4

##   
## Attaching package: 'pls'

## The following object is masked from 'package:stats':  
##   
## loadings

library(ggplot2)  
library(grid)  
library(gridExtra)

##   
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':  
##   
## combine

library(caret)

## Warning: package 'caret' was built under R version 4.0.4

## Loading required package: lattice

##   
## Attaching package: 'caret'

## The following object is masked from 'package:pls':  
##   
## R2

## The following object is masked from 'package:purrr':  
##   
## lift

library(leaps)

## Warning: package 'leaps' was built under R version 4.0.5

library(MASS)

##   
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':  
##   
## select

library(caTools)

# Data Exploration

CountyDataClean2<-read.csv("CountyDataClean2.csv",header = TRUE,sep=",")  
#names(CountyDataClean2)  
Trump16 = CountyDataClean2[, c(4, 18, 21:26, 28, 30, 32:38, 49:53)]  
Trump16$ptrump16<-Trump16$percentage16\_Donald\_Trump  
#names(Trump16)  
Trump16<-Trump16[,-1]  
  
  
  
dim(Trump16)

## [1] 3046 22

head(Trump16)

## TotalPop Hispanic White Black Native Asian Pacific Income IncomePerCap  
## 1 24788 1.3 68.9 27.6 0.1 0.3 0.0 35254 19234  
## 2 62607 2.4 77.5 17.6 0.1 0.1 0.0 40492 21591  
## 3 32840 8.8 60.3 28.3 0.3 0.7 0.0 42260 24266  
## 4 435117 7.9 85.2 1.2 0.4 2.6 0.1 60151 31642  
## 5 7192 1.7 96.6 0.3 0.0 0.4 0.0 49477 28861  
## 6 19304 1.8 93.4 3.6 0.1 0.1 0.0 36575 18408  
## Poverty ChildPoverty Professional Service Office Construction Production  
## 1 22.7 32.1 27.2 20.7 20.8 10.6 20.7  
## 2 21.5 27.6 27.6 16.9 25.7 15.0 14.8  
## 3 19.8 31.8 31.1 17.7 18.8 15.1 17.3  
## 4 11.8 13.1 43.0 16.6 25.0 6.9 8.4  
## 5 9.5 12.1 28.2 16.9 20.0 17.3 17.6  
## 6 21.5 27.1 28.5 15.9 19.7 12.2 23.8  
## SelfEmployed FamilyWork Unemployment RatioMenWomen TrumpOrClinton ptrump16  
## 1 7.8 0.1 9.4 0.9450722 Trump 0.629  
## 2 7.6 0.3 8.9 0.9458880 Trump 0.773  
## 3 7.1 0.2 5.4 0.9593103 Trump 0.545  
## 4 6.6 0.1 4.3 1.0040577 Trump 0.479  
## 5 10.4 0.5 3.0 0.9758242 Trump 0.653  
## 6 9.9 0.1 6.2 0.9958644 Trump 0.806

any(is.na(Trump16))

## [1] FALSE

colSums(is.na(Trump16))

## TotalPop Hispanic White Black Native   
## 0 0 0 0 0   
## Asian Pacific Income IncomePerCap Poverty   
## 0 0 0 0 0   
## ChildPoverty Professional Service Office Construction   
## 0 0 0 0 0   
## Production SelfEmployed FamilyWork Unemployment RatioMenWomen   
## 0 0 0 0 0   
## TrumpOrClinton ptrump16   
## 0 0

Trump16Reg <- Trump16[,c(1:20,22)]  
Trump16Class <- Trump16[,1:21]  
  
Trump20 = CountyDataClean2[, c(9,18, 21:26, 28, 30, 32:38, 49:52,54)]  
#names(Trump20)  
Trump20$ptrump20<-Trump20$percentage20\_Donald\_Trump  
Trump20<-Trump20[,-1]  
  
dim(Trump20)

## [1] 3046 22

head(Trump20)

## TotalPop Hispanic White Black Native Asian Pacific Income IncomePerCap  
## 1 24788 1.3 68.9 27.6 0.1 0.3 0.0 35254 19234  
## 2 62607 2.4 77.5 17.6 0.1 0.1 0.0 40492 21591  
## 3 32840 8.8 60.3 28.3 0.3 0.7 0.0 42260 24266  
## 4 435117 7.9 85.2 1.2 0.4 2.6 0.1 60151 31642  
## 5 7192 1.7 96.6 0.3 0.0 0.4 0.0 49477 28861  
## 6 19304 1.8 93.4 3.6 0.1 0.1 0.0 36575 18408  
## Poverty ChildPoverty Professional Service Office Construction Production  
## 1 22.7 32.1 27.2 20.7 20.8 10.6 20.7  
## 2 21.5 27.6 27.6 16.9 25.7 15.0 14.8  
## 3 19.8 31.8 31.1 17.7 18.8 15.1 17.3  
## 4 11.8 13.1 43.0 16.6 25.0 6.9 8.4  
## 5 9.5 12.1 28.2 16.9 20.0 17.3 17.6  
## 6 21.5 27.1 28.5 15.9 19.7 12.2 23.8  
## SelfEmployed FamilyWork Unemployment RatioMenWomen TrumpOrBiden ptrump20  
## 1 7.8 0.1 9.4 0.9450722 Trump 0.661  
## 2 7.6 0.3 8.9 0.9458880 Trump 0.795  
## 3 7.1 0.2 5.4 0.9593103 Trump 0.542  
## 4 6.6 0.1 4.3 1.0040577 Trump 0.504  
## 5 10.4 0.5 3.0 0.9758242 Trump 0.697  
## 6 9.9 0.1 6.2 0.9958644 Trump 0.830

any(is.na(Trump20))

## [1] FALSE

colSums(is.na(Trump20))

## TotalPop Hispanic White Black Native   
## 0 0 0 0 0   
## Asian Pacific Income IncomePerCap Poverty   
## 0 0 0 0 0   
## ChildPoverty Professional Service Office Construction   
## 0 0 0 0 0   
## Production SelfEmployed FamilyWork Unemployment RatioMenWomen   
## 0 0 0 0 0   
## TrumpOrBiden ptrump20   
## 0 0

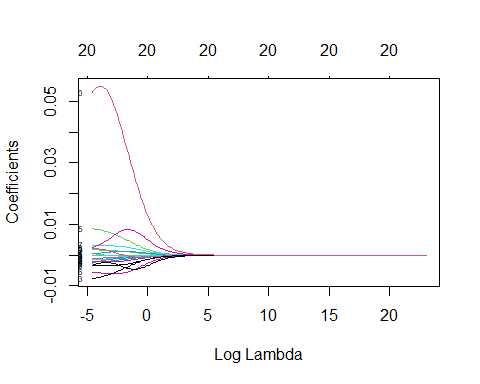
Trump20Reg <- Trump20[,c(1:20,22)]  
Trump20Class <- Trump20[,1:21]

#names(Trump16)

#Ridge Model 2016

set.seed(420)  
smp\_size <- floor(0.80 \* nrow(Trump16Reg))  
  
train <- sample(seq\_len(nrow(Trump16Reg)), size = smp\_size)  
  
training16 <- Trump16Reg[train, ]  
test16 <- Trump16Reg[-train, ]  
  
x16 <- model.matrix(ptrump16 ~ ., data = training16)[,-1]  
y16 <- training16$ptrump16

lambdaVals <- 10 ^ seq(10, -2, length = 200)  
ridge.mod16 <- glmnet(x16, y16, alpha = 0, lambda = lambdaVals)  
plot(ridge.mod16, xvar = "lambda", label = TRUE)



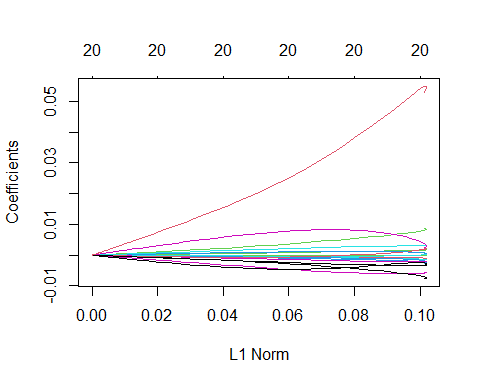
cv.ridge16 <- cv.glmnet(x16, y16, alpha = 0, lambda = 10 ^ seq(10, -15, length.out = 1000))

## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):  
## collapsing to unique 'x' values  
  
## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):  
## collapsing to unique 'x' values  
  
## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):  
## collapsing to unique 'x' values  
  
## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):  
## collapsing to unique 'x' values  
  
## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):  
## collapsing to unique 'x' values  
  
## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):  
## collapsing to unique 'x' values  
  
## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):  
## collapsing to unique 'x' values  
  
## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):  
## collapsing to unique 'x' values  
  
## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):  
## collapsing to unique 'x' values  
  
## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):  
## collapsing to unique 'x' values

cv.ridge16$lambda.min

## [1] 9.098273e-07

plot(ridge.mod16)



bestlamRidge16 <- cv.ridge16$lambda.min  
bestlamRidge16

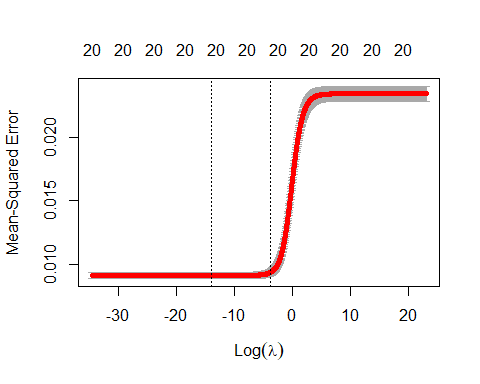
## [1] 9.098273e-07

## Estimate best ridge model using all data

coefRidge16 <- predict(glmnet(x16, y16, alpha = 0),  
 s = bestlamRidge16,  
 type = "coefficients")  
coefRidge16

## 21 x 1 sparse Matrix of class "dgCMatrix"  
## 1  
## (Intercept) 7.405575e-01  
## TotalPop -1.965711e-08  
## Hispanic -1.116493e-03  
## White 1.721644e-03  
## Black -2.312929e-03  
## Native -1.565643e-03  
## Asian -5.435580e-03  
## Pacific -3.520088e-03  
## Income 1.101120e-07  
## IncomePerCap -7.702670e-06  
## Poverty -2.734750e-03  
## ChildPoverty 7.392851e-04  
## Professional -1.850070e-03  
## Service -7.599915e-03  
## Office 2.329312e-03  
## Construction 8.713046e-03  
## Production 4.062575e-04  
## SelfEmployed 3.306404e-03  
## FamilyWork 2.252813e-03  
## Unemployment -3.371236e-03  
## RatioMenWomen 5.189714e-02

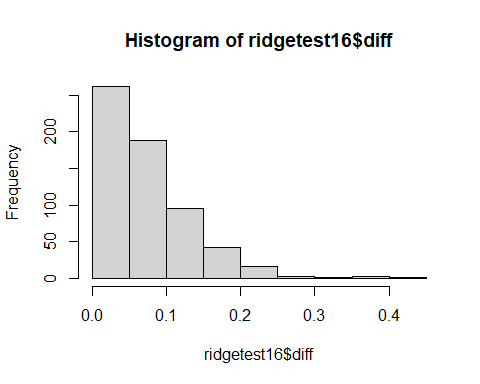
plot(cv.ridge16)



errorRidge16 <- cv.ridge16$cvm[cv.ridge16$lambda == bestlamRidge16]  
errorRidge16

## [1] 0.009136331

ridge.mod16 <- glmnet(x16, y16, alpha = 0, lambda = bestlamRidge16)  
  
testx16 <- model.matrix(ptrump16 ~ ., data = test16)[,-1]  
ridgetest16 = test16  
ridgetest16$predictions = predict.glmnet(ridge.mod16, newx = testx16, type = "response")  
ridgetest16 = ridgetest16 %>% relocate(predictions, .after = ptrump16)  
ridgetest16$diff = abs(ridgetest16$ptrump16-ridgetest16$predictions)  
ridgetest16 = ridgetest16 %>% relocate(diff, .after = predictions)  
hist(ridgetest16$diff)



MSERidge16 = mean((ridgetest16$ptrump16 - ridgetest16$predictions) ^2)  
MSERidge16

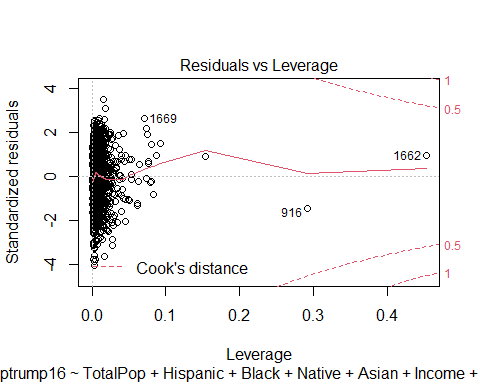
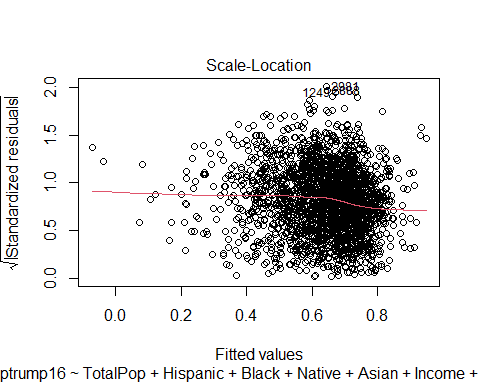
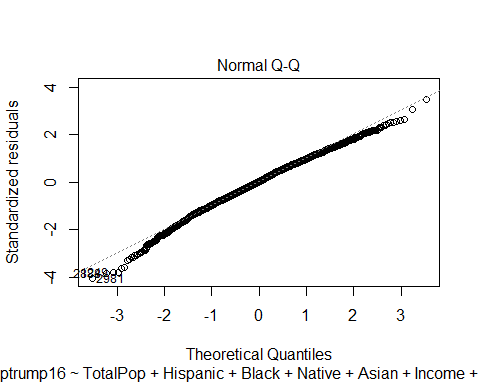
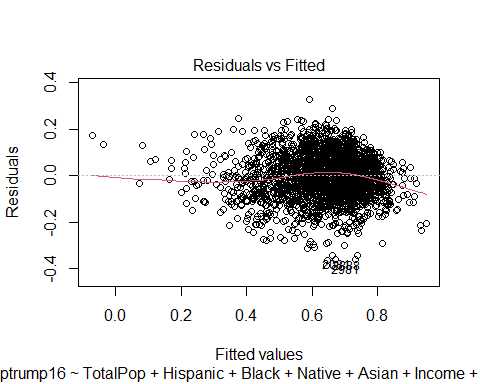
## [1] 0.008776456

#Linear Model 2016

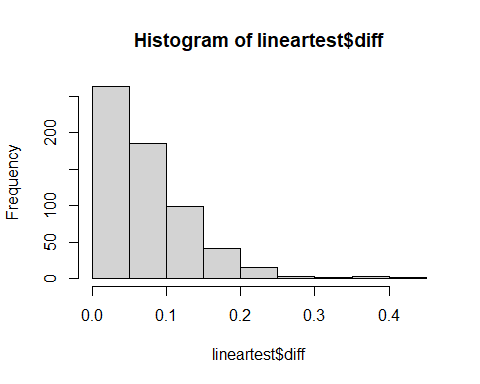
linearmodel = lm(ptrump16 ~ TotalPop + Hispanic + Black + Native + Asian + Income + IncomePerCap + Poverty + ChildPoverty + Professional + Office + Construction  
 + Production + SelfEmployed + Unemployment + RatioMenWomen, data = training16)  
  
summary(linearmodel)

##   
## Call:  
## lm(formula = ptrump16 ~ TotalPop + Hispanic + Black + Native +   
## Asian + Income + IncomePerCap + Poverty + ChildPoverty +   
## Professional + Office + Construction + Production + SelfEmployed +   
## Unemployment + RatioMenWomen, data = training16)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.38459 -0.06169 0.00250 0.06679 0.32812   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 1.535e-01 6.195e-02 2.477 0.013313 \*   
## TotalPop -1.612e-08 6.960e-09 -2.316 0.020635 \*   
## Hispanic -3.024e-03 1.665e-04 -18.158 < 2e-16 \*\*\*  
## Black -4.223e-03 1.739e-04 -24.276 < 2e-16 \*\*\*  
## Native -3.706e-03 3.432e-04 -10.799 < 2e-16 \*\*\*  
## Asian -7.815e-03 1.014e-03 -7.711 1.81e-14 \*\*\*  
## Income 9.729e-07 4.157e-07 2.341 0.019337 \*   
## IncomePerCap -1.055e-05 8.355e-07 -12.624 < 2e-16 \*\*\*  
## Poverty -4.627e-03 9.727e-04 -4.757 2.08e-06 \*\*\*  
## ChildPoverty 1.956e-03 5.297e-04 3.693 0.000227 \*\*\*  
## Professional 6.641e-03 7.286e-04 9.115 < 2e-16 \*\*\*  
## Office 1.026e-02 9.167e-04 11.194 < 2e-16 \*\*\*  
## Construction 1.689e-02 7.918e-04 21.329 < 2e-16 \*\*\*  
## Production 8.010e-03 6.451e-04 12.415 < 2e-16 \*\*\*  
## SelfEmployed 3.686e-03 6.706e-04 5.497 4.27e-08 \*\*\*  
## Unemployment -3.252e-03 9.331e-04 -3.485 0.000501 \*\*\*  
## RatioMenWomen 4.067e-02 1.627e-02 2.500 0.012485 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.09511 on 2419 degrees of freedom  
## Multiple R-squared: 0.6159, Adjusted R-squared: 0.6134   
## F-statistic: 242.5 on 16 and 2419 DF, p-value: < 2.2e-16

plot(linearmodel)



predictions = predict(linearmodel, test16)  
  
lineartest = test16  
lineartest$predictions = predictions  
lineartest = lineartest %>% relocate(predictions, .after = ptrump16)  
  
lineartest$diff = abs(lineartest$ptrump16-lineartest$predictions)  
lineartest = lineartest %>% relocate(diff, .after = predictions)  
  
hist(lineartest$diff)



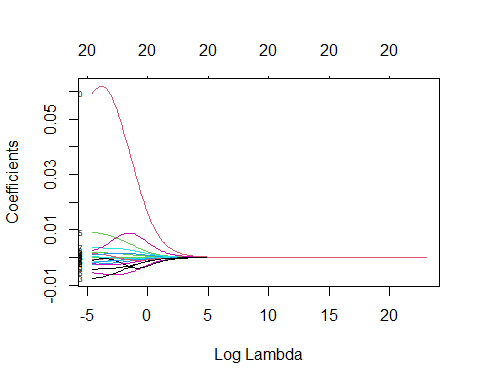
mean((lineartest$ptrump16 - lineartest$predictions)^2)

## [1] 0.008777099

#2020 ridge

set.seed(420)  
smp\_size <- floor(0.80 \* nrow(Trump20Reg))  
  
train <- sample(seq\_len(nrow(Trump20Reg)), size = smp\_size)  
  
training20 <- Trump20Reg[train, ]  
test20 <- Trump20Reg[-train, ]

x20 <- model.matrix(ptrump20 ~ ., data = training20)[,-1]  
y20 <- training20$ptrump20  
## create grid of lambda values:  
## From 10^10 down to 10^-2 = 0.1  
lambdaVals <- 10 ^ seq(10, -2, length = 200)  
ridge.mod20 <- glmnet(x20, y20, alpha = 0, lambda = lambdaVals)  
plot(ridge.mod20, xvar = "lambda", label = TRUE)



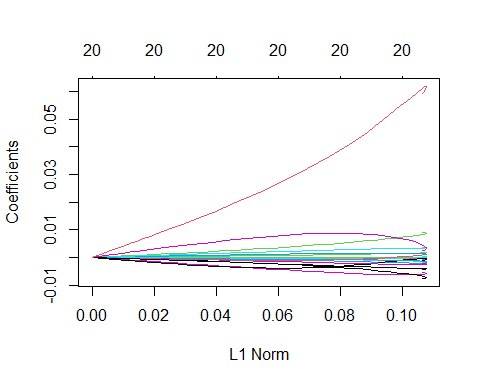
cv.ridge20 <- cv.glmnet(x20, y20, alpha = 0, lambda = 10 ^ seq(10, -15, length.out = 1000))

## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):  
## collapsing to unique 'x' values  
  
## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):  
## collapsing to unique 'x' values  
  
## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):  
## collapsing to unique 'x' values  
  
## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):  
## collapsing to unique 'x' values  
  
## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):  
## collapsing to unique 'x' values  
  
## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):  
## collapsing to unique 'x' values  
  
## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):  
## collapsing to unique 'x' values  
  
## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):  
## collapsing to unique 'x' values  
  
## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):  
## collapsing to unique 'x' values  
  
## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):  
## collapsing to unique 'x' values

cv.ridge20$lambda.min

## [1] 9.098273e-07

plot(ridge.mod20)



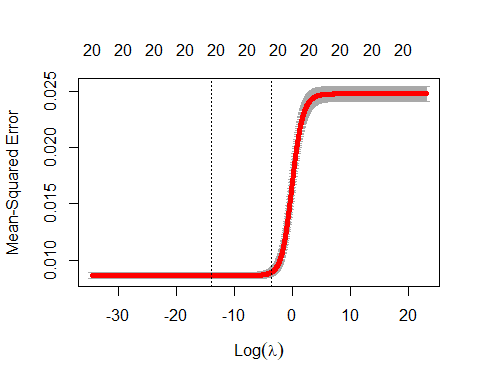
bestlamRidge20 <- cv.ridge20$lambda.min  
bestlamRidge20

## [1] 9.098273e-07

## Estimate best ridge model using all data  
coefRidge20 <- predict(glmnet(x20, y20, alpha = 0),  
 s = bestlamRidge20,  
 type = "coefficients")  
coefRidge20

## 21 x 1 sparse Matrix of class "dgCMatrix"  
## 1  
## (Intercept) 7.915202e-01  
## TotalPop -2.078191e-08  
## Hispanic -7.001473e-04  
## White 1.756006e-03  
## Black -2.544698e-03  
## Native -1.731707e-03  
## Asian -5.497280e-03  
## Pacific -1.359030e-03  
## Income 3.877360e-08  
## IncomePerCap -8.628233e-06  
## Poverty -2.372384e-03  
## ChildPoverty 6.088888e-04  
## Professional -2.217163e-03  
## Service -7.636191e-03  
## Office 1.466250e-03  
## Construction 9.118050e-03  
## Production 8.649920e-04  
## SelfEmployed 3.641471e-03  
## FamilyWork 2.111872e-03  
## Unemployment -4.231442e-03  
## RatioMenWomen 5.810760e-02

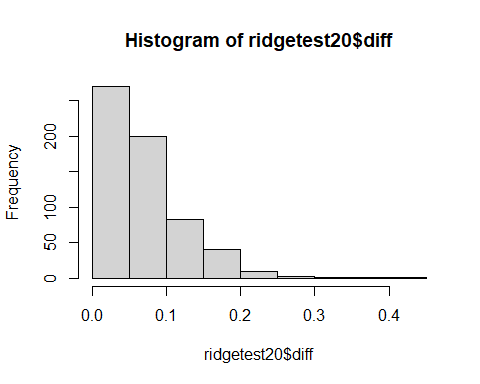
plot(cv.ridge20)



errorRidge20 <- cv.ridge20$cvm[cv.ridge20$lambda == bestlamRidge20]  
errorRidge20

## [1] 0.008634948

ridge.mod20 <- glmnet(x20, y20, alpha = 0, lambda = bestlamRidge20)  
  
testx20 <- model.matrix(ptrump20 ~ ., data = test20)[,-1]  
ridgetest20 = test20  
ridgetest20$predictions = predict.glmnet(ridge.mod20, newx = testx20, type = "response")  
ridgetest20 = ridgetest20 %>% relocate(predictions, .after = ptrump20)  
ridgetest20$diff = abs(ridgetest20$ptrump20-ridgetest20$predictions)  
ridgetest20 = ridgetest20 %>% relocate(diff, .after = predictions)  
hist(ridgetest20$diff)



MSERidge20 = mean((ridgetest20$ptrump20 - ridgetest20$predictions) ^2)  
MSERidge20

## [1] 0.007890415

coefRidge16

## 21 x 1 sparse Matrix of class "dgCMatrix"  
## 1  
## (Intercept) 7.405575e-01  
## TotalPop -1.965711e-08  
## Hispanic -1.116493e-03  
## White 1.721644e-03  
## Black -2.312929e-03  
## Native -1.565643e-03  
## Asian -5.435580e-03  
## Pacific -3.520088e-03  
## Income 1.101120e-07  
## IncomePerCap -7.702670e-06  
## Poverty -2.734750e-03  
## ChildPoverty 7.392851e-04  
## Professional -1.850070e-03  
## Service -7.599915e-03  
## Office 2.329312e-03  
## Construction 8.713046e-03  
## Production 4.062575e-04  
## SelfEmployed 3.306404e-03  
## FamilyWork 2.252813e-03  
## Unemployment -3.371236e-03  
## RatioMenWomen 5.189714e-02

coefRidge20

## 21 x 1 sparse Matrix of class "dgCMatrix"  
## 1  
## (Intercept) 7.915202e-01  
## TotalPop -2.078191e-08  
## Hispanic -7.001473e-04  
## White 1.756006e-03  
## Black -2.544698e-03  
## Native -1.731707e-03  
## Asian -5.497280e-03  
## Pacific -1.359030e-03  
## Income 3.877360e-08  
## IncomePerCap -8.628233e-06  
## Poverty -2.372384e-03  
## ChildPoverty 6.088888e-04  
## Professional -2.217163e-03  
## Service -7.636191e-03  
## Office 1.466250e-03  
## Construction 9.118050e-03  
## Production 8.649920e-04  
## SelfEmployed 3.641471e-03  
## FamilyWork 2.111872e-03  
## Unemployment -4.231442e-03  
## RatioMenWomen 5.810760e-02

str(coefRidge16)

## Formal class 'dgCMatrix' [package "Matrix"] with 6 slots  
## ..@ i : int [1:21] 0 1 2 3 4 5 6 7 8 9 ...  
## ..@ p : int [1:2] 0 21  
## ..@ Dim : int [1:2] 21 1  
## ..@ Dimnames:List of 2  
## .. ..$ : chr [1:21] "(Intercept)" "TotalPop" "Hispanic" "White" ...  
## .. ..$ : chr "1"  
## ..@ x : num [1:21] 7.41e-01 -1.97e-08 -1.12e-03 1.72e-03 -2.31e-03 ...  
## ..@ factors : list()

coefRidge16 - coefRidge20

## 21 x 1 sparse Matrix of class "dgCMatrix"  
## 1  
## (Intercept) -5.096272e-02  
## TotalPop 1.124797e-09  
## Hispanic -4.163461e-04  
## White -3.436134e-05  
## Black 2.317689e-04  
## Native 1.660646e-04  
## Asian 6.170045e-05  
## Pacific -2.161058e-03  
## Income 7.133839e-08  
## IncomePerCap 9.255633e-07  
## Poverty -3.623659e-04  
## ChildPoverty 1.303963e-04  
## Professional 3.670930e-04  
## Service 3.627612e-05  
## Office 8.630624e-04  
## Construction -4.050036e-04  
## Production -4.587344e-04  
## SelfEmployed -3.350667e-04  
## FamilyWork 1.409408e-04  
## Unemployment 8.602065e-04  
## RatioMenWomen -6.210462e-03

# Logistic Regression

# Encoding the target feature as factor  
Trump16Class$TrumpOrClinton <- as.factor(Trump16Class$TrumpOrClinton)  
Trump20Class$TrumpOrBiden <- as.factor(Trump20Class$TrumpOrBiden)

# Logistic Regression 2016

## Splitting the 2016 dataset into the Training set and Test set

set.seed(1234)  
split = sample.split(Trump16Class$TrumpOrClinton, SplitRatio = 0.80)  
training16 = subset(Trump16Class, split == TRUE)  
test16 = subset(Trump16Class, split == FALSE)  
#names(training16)  
# Feature Scaling  
training16[-21] = scale(training16[-21])  
test16[-21] = scale(test16[-21])

# Fitting Logistic Regression to the Training set

# Fitting Logistic Regression to the Training set  
logreg.model2016 = glm(formula = TrumpOrClinton ~ .,  
 family=binomial(link="logit"),  
 data = training16)

summary(logreg.model2016)

##   
## Call:  
## glm(formula = TrumpOrClinton ~ ., family = binomial(link = "logit"),   
## data = training16)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -3.2516 0.0712 0.1362 0.2982 2.6206   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 3.26512 0.14493 22.529 < 2e-16 \*\*\*  
## TotalPop -0.15136 0.12715 -1.190 0.233896   
## Hispanic 0.69662 0.81090 0.859 0.390299   
## White 2.53446 1.18239 2.144 0.032072 \*   
## Black 0.35818 0.83988 0.426 0.669767   
## Native 0.39956 0.40190 0.994 0.320142   
## Asian -0.34332 0.22361 -1.535 0.124705   
## Pacific -0.05739 0.12417 -0.462 0.643944   
## Income 0.74231 0.21263 3.491 0.000481 \*\*\*  
## IncomePerCap -1.52201 0.23227 -6.553 5.65e-11 \*\*\*  
## Poverty -0.92777 0.26324 -3.524 0.000424 \*\*\*  
## ChildPoverty 0.80354 0.23198 3.464 0.000533 \*\*\*  
## Professional 2.10360 8.29760 0.254 0.799867   
## Service 0.87562 4.73325 0.185 0.853233   
## Office 1.37103 3.92926 0.349 0.727144   
## Construction 2.09531 5.30492 0.395 0.692862   
## Production 2.32117 7.47063 0.311 0.756024   
## SelfEmployed 0.08049 0.13599 0.592 0.553942   
## FamilyWork 0.23621 0.13623 1.734 0.082930 .   
## Unemployment -0.18581 0.09940 -1.869 0.061566 .   
## RatioMenWomen -0.03869 0.09117 -0.424 0.671316   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 2023.32 on 2436 degrees of freedom  
## Residual deviance: 952.74 on 2416 degrees of freedom  
## AIC: 994.74  
##   
## Number of Fisher Scoring iterations: 7

(logreg.model2016)$coefficient

## (Intercept) TotalPop Hispanic White Black   
## 3.26512004 -0.15135720 0.69662011 2.53445975 0.35818186   
## Native Asian Pacific Income IncomePerCap   
## 0.39955894 -0.34331769 -0.05739264 0.74231217 -1.52200688   
## Poverty ChildPoverty Professional Service Office   
## -0.92776908 0.80353685 2.10359966 0.87562435 1.37102916   
## Construction Production SelfEmployed FamilyWork Unemployment   
## 2.09530567 2.32117345 0.08048823 0.23620801 -0.18581348   
## RatioMenWomen   
## -0.03868701

par(mar=c(1,1,1,1))  
graphics.off()  
plot(logreg.model2016)

Pred2016 <- ifelse(predict(logreg.model2016, newdata = test16, type = "response") > 0.5, "Trump", "Clinton")  
# summarize accuracy  
table(Pred2016, test16$TrumpOrClinton)

##   
## Pred2016 Clinton Trump  
## Clinton 58 12  
## Trump 31 508

confMat16 <-table(Pred2016, test16$TrumpOrClinton)  
confMat16

##   
## Pred2016 Clinton Trump  
## Clinton 58 12  
## Trump 31 508

Sens <- round(confMat16[2,2] / sum(confMat16[,2]), 4)  
Spec <- round(confMat16[1,1] / sum(confMat16[,1]), 4)  
Accuracy <- mean(test16$TrumpOrClinton == Pred2016)  
cat(paste("Sensitivity: ", Sens, "\n Specificity: ", Spec, "\n Accuracy: ", Accuracy, sep = ""))

## Sensitivity: 0.9769  
## Specificity: 0.6517  
## Accuracy: 0.929392446633826

# Logistic Regression 2020

## Splitting the 2020 dataset into the Training set and Test set

set.seed(1234)  
split = sample.split(Trump20Class$TrumpOrBiden, SplitRatio = 0.80)  
training20 = subset(Trump20Class, split == TRUE)  
test20 = subset(Trump20Class, split == FALSE)  
  
# Feature Scaling  
training20[-21] = scale(training20[-21])  
test20 [-21] = scale(test20 [-21])

#names(training20)

# Fitting Logistic Regression to the Training set

# Fitting Logistic Regression to the Training set  
logreg.model2020 = glm(formula = TrumpOrBiden ~ .,  
 family=binomial(link="logit"),  
 data = training20)

summary(logreg.model2020)

##   
## Call:  
## glm(formula = TrumpOrBiden ~ ., family = binomial(link = "logit"),   
## data = training20)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -3.2234 0.0724 0.1412 0.2972 3.0964   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 3.12088 0.14141 22.070 < 2e-16 \*\*\*  
## TotalPop -0.61360 0.19836 -3.093 0.00198 \*\*   
## Hispanic 0.12114 0.87513 0.138 0.88991   
## White 1.39975 1.25607 1.114 0.26511   
## Black -0.33284 0.88265 -0.377 0.70610   
## Native 0.01600 0.41869 0.038 0.96952   
## Asian -0.37761 0.25814 -1.463 0.14351   
## Pacific -0.13944 0.17229 -0.809 0.41830   
## Income 0.67476 0.21235 3.178 0.00149 \*\*   
## IncomePerCap -1.62101 0.23193 -6.989 2.76e-12 \*\*\*  
## Poverty -1.30601 0.24979 -5.228 1.71e-07 \*\*\*  
## ChildPoverty 1.19344 0.22592 5.283 1.27e-07 \*\*\*  
## Professional -8.68163 8.16488 -1.063 0.28765   
## Service -5.42077 4.69408 -1.155 0.24817   
## Office -3.85996 3.84938 -1.003 0.31598   
## Construction -4.86241 5.18023 -0.939 0.34791   
## Production -7.31831 7.27020 -1.007 0.31412   
## SelfEmployed 0.12035 0.13388 0.899 0.36868   
## FamilyWork 0.18930 0.12655 1.496 0.13468   
## Unemployment -0.23073 0.10135 -2.277 0.02281 \*   
## RatioMenWomen 0.03261 0.10130 0.322 0.74750   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 2096.19 on 2436 degrees of freedom  
## Residual deviance: 963.93 on 2416 degrees of freedom  
## AIC: 1005.9  
##   
## Number of Fisher Scoring iterations: 7

(logreg.model2020)$coefficient

## (Intercept) TotalPop Hispanic White Black   
## 3.12087822 -0.61359631 0.12113796 1.39975210 -0.33284253   
## Native Asian Pacific Income IncomePerCap   
## 0.01599672 -0.37761335 -0.13944242 0.67476011 -1.62100992   
## Poverty ChildPoverty Professional Service Office   
## -1.30600514 1.19344491 -8.68163208 -5.42077073 -3.85995851   
## Construction Production SelfEmployed FamilyWork Unemployment   
## -4.86240915 -7.31830721 0.12034948 0.18930477 -0.23073224   
## RatioMenWomen   
## 0.03261208

par(mar=c(1,1,1,1))  
graphics.off()  
plot(logreg.model2020)

Pred2020 <- ifelse(predict(logreg.model2020, newdata = test20 , type = "response") > 0.5, "Biden", "Trump")  
# summarize accuracy  
table(Pred2020, test20$TrumpOrBiden)

##   
## Pred2020 Biden Trump  
## Biden 33 494  
## Trump 61 21

confMat20 <-table(Pred2020, test20$TrumpOrBiden)  
confMat20

##   
## Pred2020 Biden Trump  
## Biden 33 494  
## Trump 61 21

Sens <- round(confMat20[2,2] / sum(confMat20[,2]), 4)  
Spec <- round(confMat20[1,1] / sum(confMat20[,1]), 4)  
Accuracy <- mean(test20$TrumpOrBiden == Pred2020)  
cat(paste("Sensitivity: ", Sens, "\n Specificity: ", Spec, "\n Accuracy: ",Accuracy, sep = ""))

## Sensitivity: 0.0408  
## Specificity: 0.3511  
## Accuracy: 0.0886699507389163

# Using LDA 2016

# Splitting the 2016 dataset into the Training set and Test set

set.seed(1234)  
split = sample.split(Trump16Class$TrumpOrClinton, SplitRatio = 0.80)  
training16 = subset(Trump16Class, split == TRUE)  
test16 = subset(Trump16Class, split == FALSE)  
#names(training16)  
# Feature Scaling  
training16[-21] = scale(training16[-21])  
test16[-21] = scale(test16[-21])

lda.model16<- lda(TrumpOrClinton~.,data=training16,family=binomial(link="logit"),)

#summary(lda.model16)  
lda.model16.pred <- predict(lda.model16, test16)  
lda.model16.confusion<-table(lda.model16.pred$class,test16$TrumpOrClinton)  
lda.model16.confusion

##   
## Clinton Trump  
## Clinton 61 12  
## Trump 28 508

Sens <- round(lda.model16.confusion[2,2] / sum(lda.model16.confusion[,2]), 4)  
Spec <- round(lda.model16.confusion[1,1] / sum(lda.model16.confusion[,1]), 4)  
Accuracy <- mean(test16$TrumpOrClinton == lda.model16.pred$class)  
cat(paste("Sensitivity: ", Sens, "\n Specificity: ", Spec, "\n Accuracy: ",  
 Accuracy, sep = ""))

## Sensitivity: 0.9769  
## Specificity: 0.6854  
## Accuracy: 0.93431855500821

# Using LDA 2020

# Splitting the 2020 dataset into the Training set and Test set

set.seed(1234)  
split = sample.split(Trump20Class$TrumpOrBiden, SplitRatio = 0.80)  
training20 = subset(Trump20Class, split == TRUE)  
test20 = subset(Trump20Class, split == FALSE)  
#names(training20)  
# Feature Scaling  
training20[-21] = scale(training20[-21])  
test20 [-21] = scale(test20 [-21])

lda.model20<- lda(TrumpOrBiden~.,data=training20,family=binomial(link="logit"),)

#summary(lda.model20)  
lda.model20.pred <- predict(lda.model20, test20 )  
lda.model20.confusion<-table(lda.model20.pred$class,test20 $TrumpOrBiden)  
lda.model20.confusion

##   
## Biden Trump  
## Biden 57 21  
## Trump 37 494

Sens <- round(lda.model20.confusion[2,2] / sum(lda.model20.confusion[,2]), 4)  
Spec <- round(lda.model20.confusion[1,1] / sum(lda.model20.confusion[,1]), 4)  
Accuracy <- mean(test20 $TrumpOrBiden == lda.model20.pred$class)  
cat(paste("Sensitivity: ", Sens, "\n Specificity: ", Spec, "\n Accuracy: ",  
 Accuracy, sep = ""))

## Sensitivity: 0.9592  
## Specificity: 0.6064  
## Accuracy: 0.904761904761905

# Using QDA 2016

# Splitting the 2016 dataset into the Training set and Test set

set.seed(1234)  
split = sample.split(Trump16Class$TrumpOrClinton, SplitRatio = 0.80)  
training16 = subset(Trump16Class , split == TRUE)  
test16 = subset(Trump16Class , split == FALSE)  
#names(training16 )  
# Feature Scaling  
training16 [-21] = scale(training16 [-21])  
test16 [-21] = scale(test16 [-21])

qda.model16<-qda(TrumpOrClinton~.,data=training16,family=binomial(link="logit"),)

#summary(qda.model16)  
qda.model16.pred <- predict(qda.model16, test16)  
qda.model16.confusion<-table(qda.model16.pred$class, test16$TrumpOrClinton)  
qda.model16.confusion

##   
## Clinton Trump  
## Clinton 65 20  
## Trump 24 500

Sens <- round(qda.model16.confusion[2,2] / sum(qda.model16.confusion[,2]), 4)  
Spec <- round(qda.model16.confusion[1,1] / sum(qda.model16.confusion[,1]), 4)  
Accuracy <- mean(test16$TrumpOrClinton == qda.model16.pred$class)  
cat(paste("Sensitivity: ", Sens, "\n Specificity: ", Spec, "\n Accuracy: ",  
 Accuracy, sep = ""))

## Sensitivity: 0.9615  
## Specificity: 0.7303  
## Accuracy: 0.927750410509031

# Using QDA 2020

# Splitting the 2020 dataset into the Training set and Test set

set.seed(1234)  
split = sample.split(Trump20Class$TrumpOrBiden, SplitRatio = 0.80)  
training20 = subset(Trump20Class, split == TRUE)  
test20 = subset(Trump20Class, split == FALSE)  
#names(training20)  
# Feature Scaling  
training20[-21] = scale(training20[-21])  
test20 [-21] = scale(test20 [-21])

qda.model20<- qda(TrumpOrBiden~.,data=training20,family=binomial(link="logit"),)

#summary(qda.model20)  
qda.model20.pred <- predict(qda.model20, test20 )  
qda.model20.confusion<-table(qda.model20.pred$class, test20$TrumpOrBiden)  
qda.model20.confusion

##   
## Biden Trump  
## Biden 47 27  
## Trump 47 488

Sens <- round(qda.model20.confusion[2,2] / sum(qda.model20.confusion[,2]), 4)  
Spec <- round(qda.model20.confusion[1,1] / sum(qda.model20.confusion[,1]), 4)  
Accuracy <- mean(test20 $TrumpOrBiden == qda.model20.pred$class)  
cat(paste("Sensitivity: ", Sens, "\n Specificity: ", Spec, "\n Accuracy: ",  
 Accuracy, sep = ""))

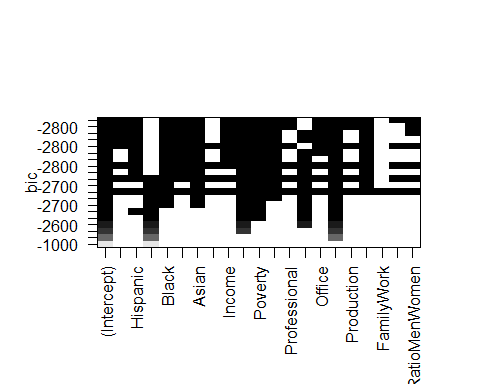
## Sensitivity: 0.9476  
## Specificity: 0.5  
## Accuracy: 0.878489326765189

# regsubset 2016

regmodel2016<-regsubsets(ptrump16 ~ ., data = Trump16Reg, nvmax = 22)  
summary(regmodel2016)$outmat

## TotalPop Hispanic White Black Native Asian Pacific Income  
## 1 ( 1 ) " " " " "\*" " " " " " " " " " "   
## 2 ( 1 ) " " " " "\*" " " " " " " " " " "   
## 3 ( 1 ) " " " " "\*" " " " " " " " " " "   
## 4 ( 1 ) " " " " "\*" " " " " " " " " " "   
## 5 ( 1 ) " " " " "\*" " " " " " " " " " "   
## 6 ( 1 ) " " "\*" "\*" " " " " " " " " " "   
## 7 ( 1 ) " " " " "\*" "\*" " " "\*" " " " "   
## 8 ( 1 ) " " " " "\*" "\*" " " "\*" " " " "   
## 9 ( 1 ) " " " " "\*" "\*" " " "\*" " " " "   
## 10 ( 1 ) " " "\*" " " "\*" "\*" "\*" " " " "   
## 11 ( 1 ) " " "\*" " " "\*" "\*" "\*" " " "\*"   
## 12 ( 1 ) " " "\*" " " "\*" "\*" "\*" " " "\*"   
## 13 ( 1 ) "\*" "\*" " " "\*" "\*" "\*" " " "\*"   
## 14 ( 1 ) "\*" "\*" " " "\*" "\*" "\*" " " "\*"   
## 15 ( 1 ) "\*" "\*" " " "\*" "\*" "\*" " " "\*"   
## 16 ( 1 ) "\*" "\*" " " "\*" "\*" "\*" " " "\*"   
## 17 ( 1 ) "\*" "\*" " " "\*" "\*" "\*" "\*" "\*"   
## 18 ( 1 ) "\*" "\*" " " "\*" "\*" "\*" "\*" "\*"   
## 19 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*"   
## 20 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*"   
## IncomePerCap Poverty ChildPoverty Professional Service Office  
## 1 ( 1 ) " " " " " " " " " " " "   
## 2 ( 1 ) " " " " " " " " " " " "   
## 3 ( 1 ) "\*" " " " " " " " " " "   
## 4 ( 1 ) "\*" " " " " " " "\*" " "   
## 5 ( 1 ) "\*" "\*" " " " " "\*" " "   
## 6 ( 1 ) "\*" "\*" " " " " "\*" " "   
## 7 ( 1 ) "\*" "\*" " " " " "\*" " "   
## 8 ( 1 ) "\*" "\*" "\*" " " "\*" " "   
## 9 ( 1 ) "\*" "\*" "\*" " " "\*" " "   
## 10 ( 1 ) "\*" "\*" "\*" " " "\*" " "   
## 11 ( 1 ) "\*" "\*" "\*" " " "\*" " "   
## 12 ( 1 ) "\*" "\*" "\*" " " "\*" "\*"   
## 13 ( 1 ) "\*" "\*" "\*" " " "\*" "\*"   
## 14 ( 1 ) "\*" "\*" "\*" " " "\*" "\*"   
## 15 ( 1 ) "\*" "\*" "\*" "\*" " " "\*"   
## 16 ( 1 ) "\*" "\*" "\*" "\*" " " "\*"   
## 17 ( 1 ) "\*" "\*" "\*" "\*" " " "\*"   
## 18 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*"   
## 19 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*"   
## 20 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*"   
## Construction Production SelfEmployed FamilyWork Unemployment  
## 1 ( 1 ) " " " " " " " " " "   
## 2 ( 1 ) "\*" " " " " " " " "   
## 3 ( 1 ) "\*" " " " " " " " "   
## 4 ( 1 ) "\*" " " " " " " " "   
## 5 ( 1 ) "\*" " " " " " " " "   
## 6 ( 1 ) "\*" " " " " " " " "   
## 7 ( 1 ) "\*" " " " " " " " "   
## 8 ( 1 ) "\*" " " " " " " " "   
## 9 ( 1 ) "\*" " " "\*" " " " "   
## 10 ( 1 ) "\*" " " "\*" " " " "   
## 11 ( 1 ) "\*" " " "\*" " " " "   
## 12 ( 1 ) "\*" " " "\*" " " " "   
## 13 ( 1 ) "\*" " " "\*" " " " "   
## 14 ( 1 ) "\*" " " "\*" " " " "   
## 15 ( 1 ) "\*" "\*" "\*" " " " "   
## 16 ( 1 ) "\*" "\*" "\*" " " "\*"   
## 17 ( 1 ) "\*" "\*" "\*" " " "\*"   
## 18 ( 1 ) "\*" "\*" "\*" " " "\*"   
## 19 ( 1 ) "\*" "\*" "\*" " " "\*"   
## 20 ( 1 ) "\*" "\*" "\*" "\*" "\*"   
## RatioMenWomen  
## 1 ( 1 ) " "   
## 2 ( 1 ) " "   
## 3 ( 1 ) " "   
## 4 ( 1 ) " "   
## 5 ( 1 ) " "   
## 6 ( 1 ) " "   
## 7 ( 1 ) " "   
## 8 ( 1 ) " "   
## 9 ( 1 ) " "   
## 10 ( 1 ) " "   
## 11 ( 1 ) " "   
## 12 ( 1 ) " "   
## 13 ( 1 ) " "   
## 14 ( 1 ) "\*"   
## 15 ( 1 ) "\*"   
## 16 ( 1 ) "\*"   
## 17 ( 1 ) "\*"   
## 18 ( 1 ) "\*"   
## 19 ( 1 ) "\*"   
## 20 ( 1 ) "\*"

par(mfrow=c(1,1))  
plot(regmodel2016)



regmodel\_Cp<-summary(regmodel2016)$cp  
regmodel\_Cp

## [1] 2581.43805 951.25625 553.86671 316.20357 207.27361 169.89034  
## [7] 131.90053 110.72833 87.46846 73.25906 60.08707 49.03387  
## [13] 38.96546 31.56511 25.05321 18.73980 17.72402 18.28801  
## [19] 19.15293 21.00000

regmodel\_bic<-summary(regmodel2016)$bic  
regmodel\_bic

## [1] -1012.718 -2048.979 -2362.109 -2564.427 -2658.908 -2688.123 -2718.384  
## [8] -2732.768 -2749.359 -2757.237 -2764.173 -2769.079 -2773.059 -2774.418  
## [15] -2774.916 -2775.238 -2770.249 -2763.673 -2756.794 -2748.926

regmodel\_adjr2<-summary(regmodel2016)$adjr2  
regmodel\_adjr2

## [1] 0.2863862 0.4933435 0.5438927 0.5741860 0.5881414 0.5930119 0.5979626  
## [8] 0.6007776 0.6038600 0.6057926 0.6075943 0.6091274 0.6105360 0.6116055  
## [15] 0.6125625 0.6134948 0.6137519 0.6138075 0.6138248 0.6137167

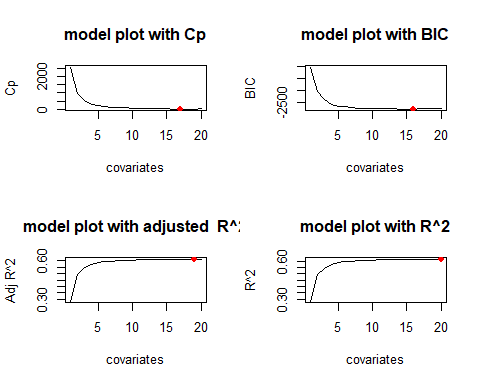
regmodel\_r2<-summary(regmodel2016)$rsq  
regmodel\_r2

## [1] 0.2866205 0.4936763 0.5443421 0.5747454 0.5888177 0.5938138 0.5988869  
## [8] 0.6018264 0.6050309 0.6070872 0.6090119 0.6106678 0.6121987 0.6133913  
## [15] 0.6144711 0.6155257 0.6159083 0.6160904 0.6162344 0.6162538

## Create data.frame of results  
regmodel2016DF <- data.frame(numVars = 1:20, Cp = regmodel\_Cp,   
 BIC = regmodel\_bic,  
 Adjr2 = regmodel\_adjr2,  
 r2 = regmodel\_r2)  
regmodel2016DF

## numVars Cp BIC Adjr2 r2  
## 1 1 2581.43805 -1012.718 0.2863862 0.2866205  
## 2 2 951.25625 -2048.979 0.4933435 0.4936763  
## 3 3 553.86671 -2362.109 0.5438927 0.5443421  
## 4 4 316.20357 -2564.427 0.5741860 0.5747454  
## 5 5 207.27361 -2658.908 0.5881414 0.5888177  
## 6 6 169.89034 -2688.123 0.5930119 0.5938138  
## 7 7 131.90053 -2718.384 0.5979626 0.5988869  
## 8 8 110.72833 -2732.768 0.6007776 0.6018264  
## 9 9 87.46846 -2749.359 0.6038600 0.6050309  
## 10 10 73.25906 -2757.237 0.6057926 0.6070872  
## 11 11 60.08707 -2764.173 0.6075943 0.6090119  
## 12 12 49.03387 -2769.079 0.6091274 0.6106678  
## 13 13 38.96546 -2773.059 0.6105360 0.6121987  
## 14 14 31.56511 -2774.418 0.6116055 0.6133913  
## 15 15 25.05321 -2774.916 0.6125625 0.6144711  
## 16 16 18.73980 -2775.238 0.6134948 0.6155257  
## 17 17 17.72402 -2770.249 0.6137519 0.6159083  
## 18 18 18.28801 -2763.673 0.6138075 0.6160904  
## 19 19 19.15293 -2756.794 0.6138248 0.6162344  
## 20 20 21.00000 -2748.926 0.6137167 0.6162538

par(mfrow=c(2,2))  
  
plot(regmodel\_Cp,main= "model plot with Cp",xlab="covariates",ylab="Cp",type="l")  
points(which.min(regmodel\_Cp),regmodel\_Cp[which.min(regmodel\_Cp)],col="red",cex=1,pch=16)  
  
plot(regmodel\_bic,main = "model plot with BIC",xlab="covariates",ylab="BIC",type="l")  
points(which.min(regmodel\_bic),regmodel\_bic[which.min(regmodel\_bic)],col="red",cex=1,pch=16)  
  
plot(regmodel\_adjr2,main = "model plot with adjusted R^2",xlab="covariates",ylab="Adj R^2",type="l")  
points(which.max(regmodel\_adjr2),regmodel\_adjr2[which.max(regmodel\_adjr2)],col="red",cex=1,pch=16)  
  
plot(regmodel\_r2,main = "model plot with R^2",xlab="covariates",ylab="R^2",type="l")  
points(which.max(regmodel\_r2),regmodel\_r2[which.max(regmodel\_r2)],col="red",cex=1,pch=16)



cp\_coef<-coef(regmodel2016,which.min(regmodel\_Cp))  
cp\_coef

## (Intercept) TotalPop Hispanic Black Native   
## 1.100142e-01 -2.127209e-08 -3.050797e-03 -4.281231e-03 -3.816592e-03   
## Asian Pacific Income IncomePerCap Poverty   
## -7.699884e-03 -9.058645e-03 1.254928e-06 -1.062097e-05 -4.474324e-03   
## ChildPoverty Professional Office Construction Production   
## 1.951059e-03 6.771497e-03 1.081365e-02 1.689208e-02 8.164305e-03   
## SelfEmployed Unemployment RatioMenWomen   
## 3.946767e-03 -2.426945e-03 4.673552e-02

bic\_coef<-coef(regmodel2016,which.min(regmodel\_bic))  
bic\_coef

## (Intercept) TotalPop Hispanic Black Native   
## 9.886696e-02 -1.955390e-08 -3.032816e-03 -4.275374e-03 -3.844078e-03   
## Asian Income IncomePerCap Poverty ChildPoverty   
## -8.660929e-03 1.261014e-06 -1.049080e-05 -4.400874e-03 1.958625e-03   
## Professional Office Construction Production SelfEmployed   
## 6.914489e-03 1.081245e-02 1.690163e-02 8.260045e-03 3.875423e-03   
## Unemployment RatioMenWomen   
## -2.384826e-03 4.722098e-02

adjr2\_coef<-coef(regmodel2016,which.max(regmodel\_adjr2))  
adjr2\_coef

## (Intercept) TotalPop Hispanic White Black   
## -2.899511e+00 -2.123220e-08 -4.358822e-03 -1.324103e-03 -5.587531e-03   
## Native Asian Pacific Income IncomePerCap   
## -5.211297e-03 -9.190117e-03 -1.210896e-02 1.242443e-06 -1.055488e-05   
## Poverty ChildPoverty Professional Service Office   
## -4.436027e-03 1.935944e-03 3.814909e-02 3.136977e-02 4.215932e-02   
## Construction Production SelfEmployed Unemployment RatioMenWomen   
## 4.829515e-02 3.958530e-02 3.960341e-03 -2.462457e-03 4.699014e-02

r2\_coef<-coef(regmodel2016,which.max(regmodel\_r2))  
r2\_coef

## (Intercept) TotalPop Hispanic White Black   
## -2.922206e+00 -2.120865e-08 -4.363675e-03 -1.326136e-03 -5.590851e-03   
## Native Asian Pacific Income IncomePerCap   
## -5.206298e-03 -9.191375e-03 -1.210731e-02 1.235830e-06 -1.055106e-05   
## Poverty ChildPoverty Professional Service Office   
## -4.427638e-03 1.931809e-03 3.838550e-02 3.159396e-02 4.237948e-02   
## Construction Production SelfEmployed FamilyWork Unemployment   
## 4.854511e-02 3.981032e-02 4.006033e-03 -1.605592e-03 -2.471051e-03   
## RatioMenWomen   
## 4.718832e-02

# forward 2016

forward.model2016<- regsubsets(ptrump16 ~ ., data = Trump16Reg, method ="forward", nvmax = 21)  
summary(forward.model2016)

## Subset selection object  
## Call: regsubsets.formula(ptrump16 ~ ., data = Trump16Reg, method = "forward",   
## nvmax = 21)  
## 20 Variables (and intercept)  
## Forced in Forced out  
## TotalPop FALSE FALSE  
## Hispanic FALSE FALSE  
## White FALSE FALSE  
## Black FALSE FALSE  
## Native FALSE FALSE  
## Asian FALSE FALSE  
## Pacific FALSE FALSE  
## Income FALSE FALSE  
## IncomePerCap FALSE FALSE  
## Poverty FALSE FALSE  
## ChildPoverty FALSE FALSE  
## Professional FALSE FALSE  
## Service FALSE FALSE  
## Office FALSE FALSE  
## Construction FALSE FALSE  
## Production FALSE FALSE  
## SelfEmployed FALSE FALSE  
## FamilyWork FALSE FALSE  
## Unemployment FALSE FALSE  
## RatioMenWomen FALSE FALSE  
## 1 subsets of each size up to 20  
## Selection Algorithm: forward  
## TotalPop Hispanic White Black Native Asian Pacific Income  
## 1 ( 1 ) " " " " "\*" " " " " " " " " " "   
## 2 ( 1 ) " " " " "\*" " " " " " " " " " "   
## 3 ( 1 ) " " " " "\*" " " " " " " " " " "   
## 4 ( 1 ) " " " " "\*" " " " " " " " " " "   
## 5 ( 1 ) " " " " "\*" " " " " " " " " " "   
## 6 ( 1 ) " " "\*" "\*" " " " " " " " " " "   
## 7 ( 1 ) " " "\*" "\*" " " " " " " " " " "   
## 8 ( 1 ) " " "\*" "\*" " " " " " " " " " "   
## 9 ( 1 ) "\*" "\*" "\*" " " " " " " " " " "   
## 10 ( 1 ) "\*" "\*" "\*" " " " " " " " " " "   
## 11 ( 1 ) "\*" "\*" "\*" " " " " " " " " "\*"   
## 12 ( 1 ) "\*" "\*" "\*" " " " " "\*" " " "\*"   
## 13 ( 1 ) "\*" "\*" "\*" " " " " "\*" " " "\*"   
## 14 ( 1 ) "\*" "\*" "\*" " " " " "\*" " " "\*"   
## 15 ( 1 ) "\*" "\*" "\*" "\*" " " "\*" " " "\*"   
## 16 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" " " "\*"   
## 17 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" " " "\*"   
## 18 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*"   
## 19 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*"   
## 20 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*"   
## IncomePerCap Poverty ChildPoverty Professional Service Office  
## 1 ( 1 ) " " " " " " " " " " " "   
## 2 ( 1 ) " " " " " " " " " " " "   
## 3 ( 1 ) "\*" " " " " " " " " " "   
## 4 ( 1 ) "\*" " " " " " " "\*" " "   
## 5 ( 1 ) "\*" "\*" " " " " "\*" " "   
## 6 ( 1 ) "\*" "\*" " " " " "\*" " "   
## 7 ( 1 ) "\*" "\*" " " " " "\*" " "   
## 8 ( 1 ) "\*" "\*" "\*" " " "\*" " "   
## 9 ( 1 ) "\*" "\*" "\*" " " "\*" " "   
## 10 ( 1 ) "\*" "\*" "\*" " " "\*" "\*"   
## 11 ( 1 ) "\*" "\*" "\*" " " "\*" "\*"   
## 12 ( 1 ) "\*" "\*" "\*" " " "\*" "\*"   
## 13 ( 1 ) "\*" "\*" "\*" " " "\*" "\*"   
## 14 ( 1 ) "\*" "\*" "\*" " " "\*" "\*"   
## 15 ( 1 ) "\*" "\*" "\*" " " "\*" "\*"   
## 16 ( 1 ) "\*" "\*" "\*" " " "\*" "\*"   
## 17 ( 1 ) "\*" "\*" "\*" " " "\*" "\*"   
## 18 ( 1 ) "\*" "\*" "\*" " " "\*" "\*"   
## 19 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*"   
## 20 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*"   
## Construction Production SelfEmployed FamilyWork Unemployment  
## 1 ( 1 ) " " " " " " " " " "   
## 2 ( 1 ) "\*" " " " " " " " "   
## 3 ( 1 ) "\*" " " " " " " " "   
## 4 ( 1 ) "\*" " " " " " " " "   
## 5 ( 1 ) "\*" " " " " " " " "   
## 6 ( 1 ) "\*" " " " " " " " "   
## 7 ( 1 ) "\*" " " "\*" " " " "   
## 8 ( 1 ) "\*" " " "\*" " " " "   
## 9 ( 1 ) "\*" " " "\*" " " " "   
## 10 ( 1 ) "\*" " " "\*" " " " "   
## 11 ( 1 ) "\*" " " "\*" " " " "   
## 12 ( 1 ) "\*" " " "\*" " " " "   
## 13 ( 1 ) "\*" " " "\*" " " " "   
## 14 ( 1 ) "\*" " " "\*" " " "\*"   
## 15 ( 1 ) "\*" " " "\*" " " "\*"   
## 16 ( 1 ) "\*" " " "\*" " " "\*"   
## 17 ( 1 ) "\*" "\*" "\*" " " "\*"   
## 18 ( 1 ) "\*" "\*" "\*" " " "\*"   
## 19 ( 1 ) "\*" "\*" "\*" " " "\*"   
## 20 ( 1 ) "\*" "\*" "\*" "\*" "\*"   
## RatioMenWomen  
## 1 ( 1 ) " "   
## 2 ( 1 ) " "   
## 3 ( 1 ) " "   
## 4 ( 1 ) " "   
## 5 ( 1 ) " "   
## 6 ( 1 ) " "   
## 7 ( 1 ) " "   
## 8 ( 1 ) " "   
## 9 ( 1 ) " "   
## 10 ( 1 ) " "   
## 11 ( 1 ) " "   
## 12 ( 1 ) " "   
## 13 ( 1 ) "\*"   
## 14 ( 1 ) "\*"   
## 15 ( 1 ) "\*"   
## 16 ( 1 ) "\*"   
## 17 ( 1 ) "\*"   
## 18 ( 1 ) "\*"   
## 19 ( 1 ) "\*"   
## 20 ( 1 ) "\*"

#plot(forward.model2016)  
  
fwdmod\_cp<-summary(forward.model2016)$cp  
fwdmod\_cp

## [1] 2581.43805 951.25625 553.86671 316.20357 207.27361 169.89034  
## [7] 139.54306 115.59212 92.60576 75.47471 64.76161 54.83957  
## [13] 48.57731 43.19051 37.25814 28.09162 21.43631 19.26589  
## [19] 19.15293 21.00000

fwdmod\_bic<-summary(forward.model2016)$bic  
fwdmod\_bic

## [1] -1012.718 -2048.979 -2362.109 -2564.427 -2658.908 -2688.123 -2711.031  
## [8] -2728.051 -2744.337 -2755.059 -2759.556 -2763.322 -2763.496 -2762.821  
## [15] -2762.708 -2765.854 -2766.517 -2762.689 -2756.794 -2748.926

fwdmod\_adjr2<-summary(forward.model2016)$adjr2  
fwdmod\_adjr2

## [1] 0.2863862 0.4933435 0.5438927 0.5741860 0.5881414 0.5930119 0.5969909  
## [8] 0.6001589 0.6032064 0.6055106 0.6069991 0.6083880 0.6093114 0.6101239  
## [15] 0.6110065 0.6123022 0.6132783 0.6136827 0.6138248 0.6137167

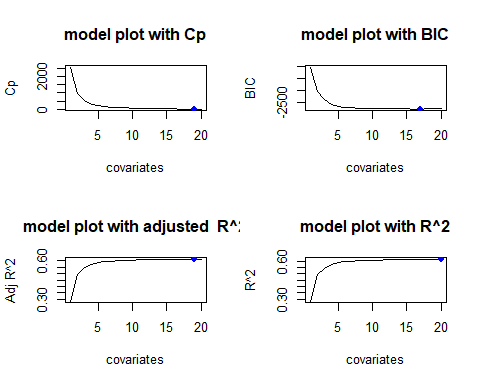
fwdmod\_r2<-summary(forward.model2016)$rsq  
fwdmod\_r2

## [1] 0.2866205 0.4936763 0.5443421 0.5747454 0.5888177 0.5938138 0.5979173  
## [8] 0.6012094 0.6043792 0.6068061 0.6084189 0.6099313 0.6109794 0.6119165  
## [15] 0.6129228 0.6143393 0.6154373 0.6159664 0.6162344 0.6162538

fwdmodel2016DF <- data.frame(numVars = 1:20, Cp = fwdmod\_cp,   
 BIC = fwdmod\_bic,  
 Adjr2 = fwdmod\_adjr2,  
 r2 = fwdmod\_r2)  
fwdmodel2016DF

## numVars Cp BIC Adjr2 r2  
## 1 1 2581.43805 -1012.718 0.2863862 0.2866205  
## 2 2 951.25625 -2048.979 0.4933435 0.4936763  
## 3 3 553.86671 -2362.109 0.5438927 0.5443421  
## 4 4 316.20357 -2564.427 0.5741860 0.5747454  
## 5 5 207.27361 -2658.908 0.5881414 0.5888177  
## 6 6 169.89034 -2688.123 0.5930119 0.5938138  
## 7 7 139.54306 -2711.031 0.5969909 0.5979173  
## 8 8 115.59212 -2728.051 0.6001589 0.6012094  
## 9 9 92.60576 -2744.337 0.6032064 0.6043792  
## 10 10 75.47471 -2755.059 0.6055106 0.6068061  
## 11 11 64.76161 -2759.556 0.6069991 0.6084189  
## 12 12 54.83957 -2763.322 0.6083880 0.6099313  
## 13 13 48.57731 -2763.496 0.6093114 0.6109794  
## 14 14 43.19051 -2762.821 0.6101239 0.6119165  
## 15 15 37.25814 -2762.708 0.6110065 0.6129228  
## 16 16 28.09162 -2765.854 0.6123022 0.6143393  
## 17 17 21.43631 -2766.517 0.6132783 0.6154373  
## 18 18 19.26589 -2762.689 0.6136827 0.6159664  
## 19 19 19.15293 -2756.794 0.6138248 0.6162344  
## 20 20 21.00000 -2748.926 0.6137167 0.6162538

par(mfrow=c(2,2))  
  
plot(fwdmod\_cp,main= "model plot with Cp",xlab="covariates",ylab="Cp",type="l")  
points(which.min(fwdmod\_cp),fwdmod\_cp[which.min(fwdmod\_cp)],col="blue",cex=1,pch=16)  
  
plot(fwdmod\_bic,main = "model plot with BIC",xlab="covariates",ylab="BIC",type="l")  
points(which.min(fwdmod\_bic),fwdmod\_bic[which.min(fwdmod\_bic)],col="blue",cex=1,pch=16)  
  
plot(fwdmod\_adjr2,main = "model plot with adjusted R^2",xlab="covariates",ylab="Adj R^2",type="l")  
points(which.max(fwdmod\_adjr2),fwdmod\_adjr2[which.max(fwdmod\_adjr2)],col="blue",cex=1,pch=16)  
  
plot(fwdmod\_r2,main = "model plot with R^2",xlab="covariates",ylab="R^2",type="l")  
points(which.max(fwdmod\_r2),fwdmod\_r2[which.max(fwdmod\_r2)],col="blue",cex=1,pch=16)



cp\_coef<-coef(forward.model2016,which.min(fwdmod\_cp))  
cp\_coef

## (Intercept) TotalPop Hispanic White Black   
## -2.899511e+00 -2.123220e-08 -4.358822e-03 -1.324103e-03 -5.587531e-03   
## Native Asian Pacific Income IncomePerCap   
## -5.211297e-03 -9.190117e-03 -1.210896e-02 1.242443e-06 -1.055488e-05   
## Poverty ChildPoverty Professional Service Office   
## -4.436027e-03 1.935944e-03 3.814909e-02 3.136977e-02 4.215932e-02   
## Construction Production SelfEmployed Unemployment RatioMenWomen   
## 4.829515e-02 3.958530e-02 3.960341e-03 -2.462457e-03 4.699014e-02

bic\_coef<-coef(forward.model2016,which.min(fwdmod\_bic))  
bic\_coef

## (Intercept) TotalPop Hispanic White Black   
## 7.989569e-01 -1.950783e-08 -3.129249e-03 -9.713443e-05 -4.372073e-03   
## Native Asian Income IncomePerCap Poverty   
## -3.946275e-03 -8.793485e-03 1.266476e-06 -1.049155e-05 -4.392741e-03   
## ChildPoverty Service Office Construction Production   
## 1.956424e-03 -6.896829e-03 3.902307e-03 9.993090e-03 1.353521e-03   
## SelfEmployed Unemployment RatioMenWomen   
## 3.882116e-03 -2.387310e-03 4.715862e-02

adjr2\_coef<-coef(forward.model2016,which.max(fwdmod\_adjr2))  
adjr2\_coef

## (Intercept) TotalPop Hispanic White Black   
## -2.899511e+00 -2.123220e-08 -4.358822e-03 -1.324103e-03 -5.587531e-03   
## Native Asian Pacific Income IncomePerCap   
## -5.211297e-03 -9.190117e-03 -1.210896e-02 1.242443e-06 -1.055488e-05   
## Poverty ChildPoverty Professional Service Office   
## -4.436027e-03 1.935944e-03 3.814909e-02 3.136977e-02 4.215932e-02   
## Construction Production SelfEmployed Unemployment RatioMenWomen   
## 4.829515e-02 3.958530e-02 3.960341e-03 -2.462457e-03 4.699014e-02

r2\_coef<-coef(forward.model2016,which.max(fwdmod\_r2))  
r2\_coef

## (Intercept) TotalPop Hispanic White Black   
## -2.922206e+00 -2.120865e-08 -4.363675e-03 -1.326136e-03 -5.590851e-03   
## Native Asian Pacific Income IncomePerCap   
## -5.206298e-03 -9.191375e-03 -1.210731e-02 1.235830e-06 -1.055106e-05   
## Poverty ChildPoverty Professional Service Office   
## -4.427638e-03 1.931809e-03 3.838550e-02 3.159396e-02 4.237948e-02   
## Construction Production SelfEmployed FamilyWork Unemployment   
## 4.854511e-02 3.981032e-02 4.006033e-03 -1.605592e-03 -2.471051e-03   
## RatioMenWomen   
## 4.718832e-02

# backward 2016

backward.model2016<-regsubsets(ptrump16 ~ ., data = Trump16Reg, method ="backward", nvmax = 21)  
summary(backward.model2016)

## Subset selection object  
## Call: regsubsets.formula(ptrump16 ~ ., data = Trump16Reg, method = "backward",   
## nvmax = 21)  
## 20 Variables (and intercept)  
## Forced in Forced out  
## TotalPop FALSE FALSE  
## Hispanic FALSE FALSE  
## White FALSE FALSE  
## Black FALSE FALSE  
## Native FALSE FALSE  
## Asian FALSE FALSE  
## Pacific FALSE FALSE  
## Income FALSE FALSE  
## IncomePerCap FALSE FALSE  
## Poverty FALSE FALSE  
## ChildPoverty FALSE FALSE  
## Professional FALSE FALSE  
## Service FALSE FALSE  
## Office FALSE FALSE  
## Construction FALSE FALSE  
## Production FALSE FALSE  
## SelfEmployed FALSE FALSE  
## FamilyWork FALSE FALSE  
## Unemployment FALSE FALSE  
## RatioMenWomen FALSE FALSE  
## 1 subsets of each size up to 20  
## Selection Algorithm: backward  
## TotalPop Hispanic White Black Native Asian Pacific Income  
## 1 ( 1 ) " " " " " " " " " " " " " " " "   
## 2 ( 1 ) " " " " " " "\*" " " " " " " " "   
## 3 ( 1 ) " " "\*" " " "\*" " " " " " " " "   
## 4 ( 1 ) " " "\*" " " "\*" " " " " " " " "   
## 5 ( 1 ) " " "\*" " " "\*" "\*" " " " " " "   
## 6 ( 1 ) " " "\*" " " "\*" "\*" " " " " " "   
## 7 ( 1 ) " " "\*" " " "\*" "\*" " " " " " "   
## 8 ( 1 ) " " "\*" " " "\*" "\*" " " " " " "   
## 9 ( 1 ) " " "\*" " " "\*" "\*" "\*" " " " "   
## 10 ( 1 ) " " "\*" " " "\*" "\*" "\*" " " " "   
## 11 ( 1 ) " " "\*" " " "\*" "\*" "\*" " " " "   
## 12 ( 1 ) " " "\*" " " "\*" "\*" "\*" " " "\*"   
## 13 ( 1 ) " " "\*" " " "\*" "\*" "\*" " " "\*"   
## 14 ( 1 ) " " "\*" " " "\*" "\*" "\*" " " "\*"   
## 15 ( 1 ) "\*" "\*" " " "\*" "\*" "\*" " " "\*"   
## 16 ( 1 ) "\*" "\*" " " "\*" "\*" "\*" " " "\*"   
## 17 ( 1 ) "\*" "\*" " " "\*" "\*" "\*" "\*" "\*"   
## 18 ( 1 ) "\*" "\*" " " "\*" "\*" "\*" "\*" "\*"   
## 19 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*"   
## 20 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*"   
## IncomePerCap Poverty ChildPoverty Professional Service Office  
## 1 ( 1 ) " " " " " " " " " " " "   
## 2 ( 1 ) " " " " " " " " " " " "   
## 3 ( 1 ) " " " " " " " " " " " "   
## 4 ( 1 ) "\*" " " " " " " " " " "   
## 5 ( 1 ) "\*" " " " " " " " " " "   
## 6 ( 1 ) "\*" " " " " " " " " " "   
## 7 ( 1 ) "\*" " " " " " " " " "\*"   
## 8 ( 1 ) "\*" " " " " "\*" " " "\*"   
## 9 ( 1 ) "\*" " " " " "\*" " " "\*"   
## 10 ( 1 ) "\*" "\*" " " "\*" " " "\*"   
## 11 ( 1 ) "\*" "\*" " " "\*" " " "\*"   
## 12 ( 1 ) "\*" "\*" " " "\*" " " "\*"   
## 13 ( 1 ) "\*" "\*" "\*" "\*" " " "\*"   
## 14 ( 1 ) "\*" "\*" "\*" "\*" " " "\*"   
## 15 ( 1 ) "\*" "\*" "\*" "\*" " " "\*"   
## 16 ( 1 ) "\*" "\*" "\*" "\*" " " "\*"   
## 17 ( 1 ) "\*" "\*" "\*" "\*" " " "\*"   
## 18 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*"   
## 19 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*"   
## 20 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*"   
## Construction Production SelfEmployed FamilyWork Unemployment  
## 1 ( 1 ) "\*" " " " " " " " "   
## 2 ( 1 ) "\*" " " " " " " " "   
## 3 ( 1 ) "\*" " " " " " " " "   
## 4 ( 1 ) "\*" " " " " " " " "   
## 5 ( 1 ) "\*" " " " " " " " "   
## 6 ( 1 ) "\*" "\*" " " " " " "   
## 7 ( 1 ) "\*" "\*" " " " " " "   
## 8 ( 1 ) "\*" "\*" " " " " " "   
## 9 ( 1 ) "\*" "\*" " " " " " "   
## 10 ( 1 ) "\*" "\*" " " " " " "   
## 11 ( 1 ) "\*" "\*" "\*" " " " "   
## 12 ( 1 ) "\*" "\*" "\*" " " " "   
## 13 ( 1 ) "\*" "\*" "\*" " " " "   
## 14 ( 1 ) "\*" "\*" "\*" " " " "   
## 15 ( 1 ) "\*" "\*" "\*" " " " "   
## 16 ( 1 ) "\*" "\*" "\*" " " "\*"   
## 17 ( 1 ) "\*" "\*" "\*" " " "\*"   
## 18 ( 1 ) "\*" "\*" "\*" " " "\*"   
## 19 ( 1 ) "\*" "\*" "\*" " " "\*"   
## 20 ( 1 ) "\*" "\*" "\*" "\*" "\*"   
## RatioMenWomen  
## 1 ( 1 ) " "   
## 2 ( 1 ) " "   
## 3 ( 1 ) " "   
## 4 ( 1 ) " "   
## 5 ( 1 ) " "   
## 6 ( 1 ) " "   
## 7 ( 1 ) " "   
## 8 ( 1 ) " "   
## 9 ( 1 ) " "   
## 10 ( 1 ) " "   
## 11 ( 1 ) " "   
## 12 ( 1 ) " "   
## 13 ( 1 ) " "   
## 14 ( 1 ) "\*"   
## 15 ( 1 ) "\*"   
## 16 ( 1 ) "\*"   
## 17 ( 1 ) "\*"   
## 18 ( 1 ) "\*"   
## 19 ( 1 ) "\*"   
## 20 ( 1 ) "\*"

#plot(backward.model2016)  
  
bwdmod\_cp<-summary(backward.model2016)$cp  
bwdmod\_cp

## [1] 3239.56159 2338.06173 1403.20355 1003.83204 644.53598 559.51013  
## [7] 488.50089 332.83657 162.34728 110.81171 72.20207 57.04278  
## [13] 42.93663 33.55996 25.05321 18.73980 17.72402 18.28801  
## [19] 19.15293 21.00000

bwdmod\_bic<-summary(backward.model2016)$bic  
bwdmod\_bic

## [1] -675.6012 -1140.5947 -1715.5855 -1996.0880 -2273.4409 -2338.3469  
## [7] -2392.8833 -2524.5058 -2676.9696 -2720.5277 -2752.2230 -2761.1404  
## [13] -2769.1044 -2772.4251 -2774.9161 -2775.2383 -2770.2492 -2763.6726  
## [19] -2756.7937 -2748.9262

bwdmod\_adjr2<-summary(backward.model2016)$adjr2  
bwdmod\_adjr2

## [1] 0.2028704 0.3173001 0.4360411 0.4868399 0.5325799 0.5434878 0.5526207  
## [8] 0.5725271 0.5943329 0.6010130 0.6060518 0.6081074 0.6100301 0.6113513  
## [15] 0.6125625 0.6134948 0.6137519 0.6138075 0.6138248 0.6137167

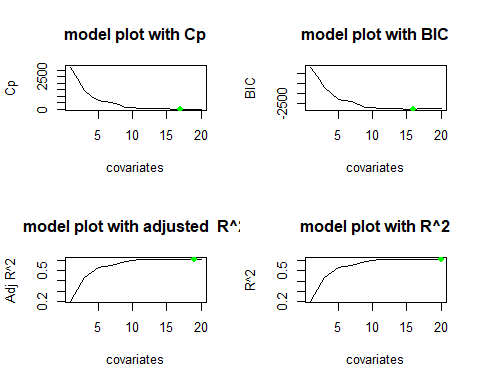
bwdmod\_r2<-summary(backward.model2016)$rsq  
bwdmod\_r2

## [1] 0.2031322 0.3177486 0.4365967 0.4875140 0.5333474 0.5443873 0.5536492  
## [8] 0.5736502 0.5955319 0.6023233 0.6074750 0.6096518 0.6116950 0.6131382  
## [15] 0.6144711 0.6155257 0.6159083 0.6160904 0.6162344 0.6162538

bwdmodel2016DF <- data.frame(numVars = 1:20, Cp = bwdmod\_cp,   
 BIC = bwdmod\_bic,  
 Adjr2 = bwdmod\_adjr2,  
 r2 = bwdmod\_r2)  
bwdmodel2016DF

## numVars Cp BIC Adjr2 r2  
## 1 1 3239.56159 -675.6012 0.2028704 0.2031322  
## 2 2 2338.06173 -1140.5947 0.3173001 0.3177486  
## 3 3 1403.20355 -1715.5855 0.4360411 0.4365967  
## 4 4 1003.83204 -1996.0880 0.4868399 0.4875140  
## 5 5 644.53598 -2273.4409 0.5325799 0.5333474  
## 6 6 559.51013 -2338.3469 0.5434878 0.5443873  
## 7 7 488.50089 -2392.8833 0.5526207 0.5536492  
## 8 8 332.83657 -2524.5058 0.5725271 0.5736502  
## 9 9 162.34728 -2676.9696 0.5943329 0.5955319  
## 10 10 110.81171 -2720.5277 0.6010130 0.6023233  
## 11 11 72.20207 -2752.2230 0.6060518 0.6074750  
## 12 12 57.04278 -2761.1404 0.6081074 0.6096518  
## 13 13 42.93663 -2769.1044 0.6100301 0.6116950  
## 14 14 33.55996 -2772.4251 0.6113513 0.6131382  
## 15 15 25.05321 -2774.9161 0.6125625 0.6144711  
## 16 16 18.73980 -2775.2383 0.6134948 0.6155257  
## 17 17 17.72402 -2770.2492 0.6137519 0.6159083  
## 18 18 18.28801 -2763.6726 0.6138075 0.6160904  
## 19 19 19.15293 -2756.7937 0.6138248 0.6162344  
## 20 20 21.00000 -2748.9262 0.6137167 0.6162538

par(mfrow=c(2,2))  
  
plot(bwdmod\_cp,main= "model plot with Cp",xlab="covariates",ylab="Cp",type="l")  
points(which.min(bwdmod\_cp),bwdmod\_cp[which.min(bwdmod\_cp)],col="green",cex=1,pch=16)  
  
plot(bwdmod\_bic,main = "model plot with BIC",xlab="covariates",ylab="BIC",type="l")  
points(which.min(bwdmod\_bic),bwdmod\_bic[which.min(bwdmod\_bic)],col="green",cex=1,pch=16)  
  
plot(bwdmod\_adjr2,main = "model plot with adjusted R^2",xlab="covariates",ylab="Adj R^2",type="l")  
points(which.max(bwdmod\_adjr2),bwdmod\_adjr2[which.max(bwdmod\_adjr2)],col="green",cex=1,pch=16)  
  
plot(bwdmod\_r2,main = "model plot with R^2",xlab="covariates",ylab="R^2",type="l")  
points(which.max(bwdmod\_r2),bwdmod\_r2[which.max(bwdmod\_r2)],col="green",cex=1,pch=16)



cp\_coef<-coef(backward.model2016,which.min(bwdmod\_cp))  
bic\_coef<-coef(backward.model2016,which.min(bwdmod\_bic))  
adjr2\_coef<-coef(backward.model2016,which.max(bwdmod\_adjr2))  
r2\_coef<-coef(backward.model2016,which.max(bwdmod\_r2))  
cp\_coef

## (Intercept) TotalPop Hispanic Black Native   
## 1.100142e-01 -2.127209e-08 -3.050797e-03 -4.281231e-03 -3.816592e-03   
## Asian Pacific Income IncomePerCap Poverty   
## -7.699884e-03 -9.058645e-03 1.254928e-06 -1.062097e-05 -4.474324e-03   
## ChildPoverty Professional Office Construction Production   
## 1.951059e-03 6.771497e-03 1.081365e-02 1.689208e-02 8.164305e-03   
## SelfEmployed Unemployment RatioMenWomen   
## 3.946767e-03 -2.426945e-03 4.673552e-02

bic\_coef

## (Intercept) TotalPop Hispanic Black Native   
## 9.886696e-02 -1.955390e-08 -3.032816e-03 -4.275374e-03 -3.844078e-03   
## Asian Income IncomePerCap Poverty ChildPoverty   
## -8.660929e-03 1.261014e-06 -1.049080e-05 -4.400874e-03 1.958625e-03   
## Professional Office Construction Production SelfEmployed   
## 6.914489e-03 1.081245e-02 1.690163e-02 8.260045e-03 3.875423e-03   
## Unemployment RatioMenWomen   
## -2.384826e-03 4.722098e-02

adjr2\_coef

## (Intercept) TotalPop Hispanic White Black   
## -2.899511e+00 -2.123220e-08 -4.358822e-03 -1.324103e-03 -5.587531e-03   
## Native Asian Pacific Income IncomePerCap   
## -5.211297e-03 -9.190117e-03 -1.210896e-02 1.242443e-06 -1.055488e-05   
## Poverty ChildPoverty Professional Service Office   
## -4.436027e-03 1.935944e-03 3.814909e-02 3.136977e-02 4.215932e-02   
## Construction Production SelfEmployed Unemployment RatioMenWomen   
## 4.829515e-02 3.958530e-02 3.960341e-03 -2.462457e-03 4.699014e-02

r2\_coef

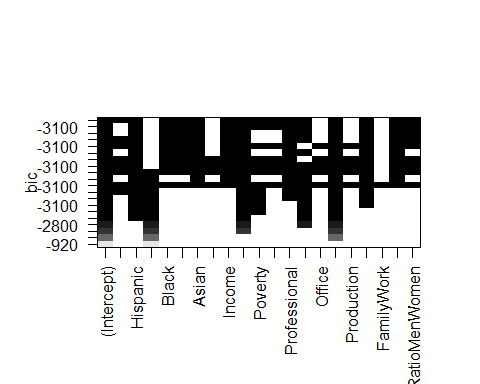
## (Intercept) TotalPop Hispanic White Black   
## -2.922206e+00 -2.120865e-08 -4.363675e-03 -1.326136e-03 -5.590851e-03   
## Native Asian Pacific Income IncomePerCap   
## -5.206298e-03 -9.191375e-03 -1.210731e-02 1.235830e-06 -1.055106e-05   
## Poverty ChildPoverty Professional Service Office   
## -4.427638e-03 1.931809e-03 3.838550e-02 3.159396e-02 4.237948e-02   
## Construction Production SelfEmployed FamilyWork Unemployment   
## 4.854511e-02 3.981032e-02 4.006033e-03 -1.605592e-03 -2.471051e-03   
## RatioMenWomen   
## 4.718832e-02

# regsubset 2020

regmodel2020<-regsubsets(ptrump20 ~ ., data = Trump20Reg, nvmax = 22)  
summary(regmodel2020)$outmat

## TotalPop Hispanic White Black Native Asian Pacific Income  
## 1 ( 1 ) " " " " "\*" " " " " " " " " " "   
## 2 ( 1 ) " " " " "\*" " " " " " " " " " "   
## 3 ( 1 ) " " " " "\*" " " " " " " " " " "   
## 4 ( 1 ) " " " " "\*" " " " " " " " " " "   
## 5 ( 1 ) " " "\*" "\*" " " " " " " " " " "   
## 6 ( 1 ) " " "\*" "\*" " " " " " " " " " "   
## 7 ( 1 ) " " "\*" "\*" " " " " " " " " " "   
## 8 ( 1 ) " " "\*" "\*" " " " " " " " " " "   
## 9 ( 1 ) "\*" "\*" "\*" " " " " " " " " " "   
## 10 ( 1 ) " " "\*" "\*" " " " " "\*" " " "\*"   
## 11 ( 1 ) " " "\*" " " "\*" "\*" "\*" " " "\*"   
## 12 ( 1 ) " " "\*" " " "\*" "\*" "\*" " " "\*"   
## 13 ( 1 ) "\*" "\*" " " "\*" "\*" "\*" " " "\*"   
## 14 ( 1 ) " " "\*" " " "\*" "\*" "\*" " " "\*"   
## 15 ( 1 ) "\*" "\*" " " "\*" "\*" "\*" " " "\*"   
## 16 ( 1 ) "\*" "\*" " " "\*" "\*" "\*" " " "\*"   
## 17 ( 1 ) "\*" "\*" " " "\*" "\*" "\*" "\*" "\*"   
## 18 ( 1 ) "\*" "\*" " " "\*" "\*" "\*" "\*" "\*"   
## 19 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*"   
## 20 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*"   
## IncomePerCap Poverty ChildPoverty Professional Service Office  
## 1 ( 1 ) " " " " " " " " " " " "   
## 2 ( 1 ) " " " " " " " " " " " "   
## 3 ( 1 ) "\*" " " " " " " " " " "   
## 4 ( 1 ) "\*" " " " " " " "\*" " "   
## 5 ( 1 ) "\*" " " " " " " "\*" " "   
## 6 ( 1 ) "\*" "\*" " " " " "\*" " "   
## 7 ( 1 ) "\*" "\*" " " " " "\*" " "   
## 8 ( 1 ) "\*" "\*" " " "\*" "\*" " "   
## 9 ( 1 ) "\*" "\*" " " "\*" "\*" " "   
## 10 ( 1 ) "\*" " " " " "\*" "\*" " "   
## 11 ( 1 ) "\*" " " " " "\*" "\*" " "   
## 12 ( 1 ) "\*" " " " " "\*" "\*" " "   
## 13 ( 1 ) "\*" " " " " "\*" "\*" " "   
## 14 ( 1 ) "\*" "\*" "\*" "\*" "\*" " "   
## 15 ( 1 ) "\*" "\*" "\*" "\*" "\*" " "   
## 16 ( 1 ) "\*" "\*" "\*" "\*" " " "\*"   
## 17 ( 1 ) "\*" "\*" "\*" "\*" " " "\*"   
## 18 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*"   
## 19 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*"   
## 20 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*"   
## Construction Production SelfEmployed FamilyWork Unemployment  
## 1 ( 1 ) " " " " " " " " " "   
## 2 ( 1 ) "\*" " " " " " " " "   
## 3 ( 1 ) "\*" " " " " " " " "   
## 4 ( 1 ) "\*" " " " " " " " "   
## 5 ( 1 ) "\*" " " " " " " " "   
## 6 ( 1 ) "\*" " " " " " " " "   
## 7 ( 1 ) "\*" " " "\*" " " " "   
## 8 ( 1 ) "\*" " " "\*" " " " "   
## 9 ( 1 ) "\*" " " "\*" " " " "   
## 10 ( 1 ) "\*" " " "\*" " " "\*"   
## 11 ( 1 ) "\*" " " "\*" " " "\*"   
## 12 ( 1 ) "\*" " " "\*" " " "\*"   
## 13 ( 1 ) "\*" " " "\*" " " "\*"   
## 14 ( 1 ) "\*" " " "\*" " " "\*"   
## 15 ( 1 ) "\*" " " "\*" " " "\*"   
## 16 ( 1 ) "\*" "\*" "\*" " " "\*"   
## 17 ( 1 ) "\*" "\*" "\*" " " "\*"   
## 18 ( 1 ) "\*" "\*" "\*" " " "\*"   
## 19 ( 1 ) "\*" "\*" "\*" " " "\*"   
## 20 ( 1 ) "\*" "\*" "\*" "\*" "\*"   
## RatioMenWomen  
## 1 ( 1 ) " "   
## 2 ( 1 ) " "   
## 3 ( 1 ) " "   
## 4 ( 1 ) " "   
## 5 ( 1 ) " "   
## 6 ( 1 ) " "   
## 7 ( 1 ) " "   
## 8 ( 1 ) " "   
## 9 ( 1 ) " "   
## 10 ( 1 ) " "   
## 11 ( 1 ) " "   
## 12 ( 1 ) "\*"   
## 13 ( 1 ) "\*"   
## 14 ( 1 ) "\*"   
## 15 ( 1 ) "\*"   
## 16 ( 1 ) "\*"   
## 17 ( 1 ) "\*"   
## 18 ( 1 ) "\*"   
## 19 ( 1 ) "\*"   
## 20 ( 1 ) "\*"

par(mfrow=c(1,1))  
plot(regmodel2020)



regmodel\_Cp<-summary(regmodel2020)$cp  
regmodel\_Cp

## [1] 3476.89318 1307.30660 691.77965 450.80934 267.60179 195.10283  
## [7] 152.09657 122.28018 99.05314 74.94391 50.78090 38.85668  
## [13] 33.03190 24.69312 17.21838 15.04657 15.94391 17.27335  
## [19] 19.03317 21.00000

regmodel\_bic<-summary(regmodel2020)$bic  
regmodel\_bic

## [1] -918.1843 -2144.2439 -2602.8918 -2800.0551 -2958.2822 -3019.7794  
## [7] -3054.5374 -3077.1247 -3093.5932 -3111.1273 -3128.9309 -3134.7436  
## [13] -3134.5239 -3136.8415 -3138.3261 -3134.4995 -3127.5877 -3120.2412  
## [19] -3112.4614 -3104.4733

regmodel\_adjr2<-summary(regmodel2020)$adjr2  
regmodel\_adjr2

## [1] 0.2638916 0.5089442 0.5785592 0.6058837 0.6266951 0.6349985 0.6399704  
## [8] 0.6434529 0.6461915 0.6490319 0.6518803 0.6533436 0.6541163 0.6551746  
## [15] 0.6561356 0.6564955 0.6565073 0.6564699 0.6563837 0.6562739

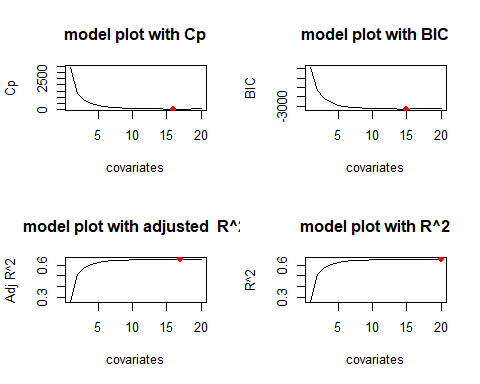
regmodel\_r2<-summary(regmodel2020)$rsq  
regmodel\_r2

## [1] 0.2641333 0.5092667 0.5789745 0.6064015 0.6273081 0.6357177 0.6407981  
## [8] 0.6443896 0.6472373 0.6501846 0.6531379 0.6547097 0.6555930 0.6567600  
## [15] 0.6578296 0.6583005 0.6584250 0.6585006 0.6585278 0.6585315

## Create data.frame of results  
regmodel2020DF <- data.frame(numVars = 1:20, Cp = regmodel\_Cp,   
 BIC = regmodel\_bic,  
 Adjr2 = regmodel\_adjr2,  
 r2 = regmodel\_r2)  
regmodel2020DF

## numVars Cp BIC Adjr2 r2  
## 1 1 3476.89318 -918.1843 0.2638916 0.2641333  
## 2 2 1307.30660 -2144.2439 0.5089442 0.5092667  
## 3 3 691.77965 -2602.8918 0.5785592 0.5789745  
## 4 4 450.80934 -2800.0551 0.6058837 0.6064015  
## 5 5 267.60179 -2958.2822 0.6266951 0.6273081  
## 6 6 195.10283 -3019.7794 0.6349985 0.6357177  
## 7 7 152.09657 -3054.5374 0.6399704 0.6407981  
## 8 8 122.28018 -3077.1247 0.6434529 0.6443896  
## 9 9 99.05314 -3093.5932 0.6461915 0.6472373  
## 10 10 74.94391 -3111.1273 0.6490319 0.6501846  
## 11 11 50.78090 -3128.9309 0.6518803 0.6531379  
## 12 12 38.85668 -3134.7436 0.6533436 0.6547097  
## 13 13 33.03190 -3134.5239 0.6541163 0.6555930  
## 14 14 24.69312 -3136.8415 0.6551746 0.6567600  
## 15 15 17.21838 -3138.3261 0.6561356 0.6578296  
## 16 16 15.04657 -3134.4995 0.6564955 0.6583005  
## 17 17 15.94391 -3127.5877 0.6565073 0.6584250  
## 18 18 17.27335 -3120.2412 0.6564699 0.6585006  
## 19 19 19.03317 -3112.4614 0.6563837 0.6585278  
## 20 20 21.00000 -3104.4733 0.6562739 0.6585315

par(mfrow=c(2,2))  
plot(regmodel\_Cp,main= "model plot with Cp",xlab="covariates",ylab="Cp",type="l")  
points(which.min(regmodel\_Cp),regmodel\_Cp[which.min(regmodel\_Cp)],col="red",cex=1,pch=16)  
  
plot(regmodel\_bic,main = "model plot with BIC",xlab="covariates",ylab="BIC",type="l")  
points(which.min(regmodel\_bic),regmodel\_bic[which.min(regmodel\_bic)],col="red",cex=1,pch=16)  
  
plot(regmodel\_adjr2,main = "model plot with adjusted R^2",xlab="covariates",ylab="Adj R^2",type="l")  
points(which.max(regmodel\_adjr2),regmodel\_adjr2[which.max(regmodel\_adjr2)],col="red",cex=1,pch=16)  
  
plot(regmodel\_r2,main = "model plot with R^2",xlab="covariates",ylab="R^2",type="l")  
points(which.max(regmodel\_r2),regmodel\_r2[which.max(regmodel\_r2)],col="red",cex=1,pch=16)



cp\_coef<-coef(regmodel2020,which.min(regmodel\_Cp))  
cp\_coef

## (Intercept) TotalPop Hispanic Black Native   
## 1.748871e-01 -1.939720e-08 -2.629746e-03 -4.536607e-03 -4.061915e-03   
## Asian Income IncomePerCap Poverty ChildPoverty   
## -8.722826e-03 1.335012e-06 -1.166245e-05 -3.686704e-03 1.653781e-03   
## Professional Office Construction Production SelfEmployed   
## 6.286753e-03 9.701021e-03 1.698839e-02 8.517806e-03 4.291127e-03   
## Unemployment RatioMenWomen   
## -3.407008e-03 4.974542e-02

bic\_coef<-coef(regmodel2020,which.min(regmodel\_bic))  
bic\_coef

## (Intercept) TotalPop Hispanic Black Native   
## 1.064448e+00 -1.848762e-08 -2.610360e-03 -4.534888e-03 -4.068025e-03   
## Asian Income IncomePerCap Poverty ChildPoverty   
## -8.807603e-03 1.346736e-06 -1.162797e-05 -3.770821e-03 1.679668e-03   
## Professional Service Construction SelfEmployed Unemployment   
## -2.355005e-03 -8.630677e-03 8.121530e-03 4.166458e-03 -3.281372e-03   
## RatioMenWomen   
## 4.743591e-02

adjr2\_coef<-coef(regmodel2020,which.max(regmodel\_adjr2))  
adjr2\_coef

## (Intercept) TotalPop Hispanic Black Native   
## 1.814060e-01 -2.040199e-08 -2.640261e-03 -4.540032e-03 -4.045842e-03   
## Asian Pacific Income IncomePerCap Poverty   
## -8.160812e-03 -5.297454e-03 1.331453e-06 -1.173858e-05 -3.729657e-03   
## ChildPoverty Professional Office Construction Production   
## 1.649357e-03 6.203132e-03 9.701727e-03 1.698281e-02 8.461818e-03   
## SelfEmployed Unemployment RatioMenWomen   
## 4.332849e-03 -3.431639e-03 4.946153e-02

r2\_coef<-coef(regmodel2020,which.max(regmodel\_r2))  
r2\_coef

## (Intercept) TotalPop Hispanic White Black   
## -1.846921e+00 -2.036460e-08 -3.224345e-03 -5.899842e-04 -5.121689e-03   
## Native Asian Pacific Income IncomePerCap   
## -4.664813e-03 -8.825566e-03 -6.646815e-03 1.320777e-06 -1.169903e-05   
## Poverty ChildPoverty Professional Service Office   
## -3.707242e-03 1.639823e-03 2.705983e-02 2.084868e-02 3.053541e-02   
## Construction Production SelfEmployed FamilyWork Unemployment   
## 3.785734e-02 2.933447e-02 4.357444e-03 -7.231303e-04 -3.453703e-03   
## RatioMenWomen   
## 4.967528e-02

# forward 2020

forward.model2020<- regsubsets( ptrump20 ~ ., data = Trump20Reg, method ="forward", nvmax = 21)  
summary(forward.model2020)

## Subset selection object  
## Call: regsubsets.formula(ptrump20 ~ ., data = Trump20Reg, method = "forward",   
## nvmax = 21)  
## 20 Variables (and intercept)  
## Forced in Forced out  
## TotalPop FALSE FALSE  
## Hispanic FALSE FALSE  
## White FALSE FALSE  
## Black FALSE FALSE  
## Native FALSE FALSE  
## Asian FALSE FALSE  
## Pacific FALSE FALSE  
## Income FALSE FALSE  
## IncomePerCap FALSE FALSE  
## Poverty FALSE FALSE  
## ChildPoverty FALSE FALSE  
## Professional FALSE FALSE  
## Service FALSE FALSE  
## Office FALSE FALSE  
## Construction FALSE FALSE  
## Production FALSE FALSE  
## SelfEmployed FALSE FALSE  
## FamilyWork FALSE FALSE  
## Unemployment FALSE FALSE  
## RatioMenWomen FALSE FALSE  
## 1 subsets of each size up to 20  
## Selection Algorithm: forward  
## TotalPop Hispanic White Black Native Asian Pacific Income  
## 1 ( 1 ) " " " " "\*" " " " " " " " " " "   
## 2 ( 1 ) " " " " "\*" " " " " " " " " " "   
## 3 ( 1 ) " " " " "\*" " " " " " " " " " "   
## 4 ( 1 ) " " " " "\*" " " " " " " " " " "   
## 5 ( 1 ) " " "\*" "\*" " " " " " " " " " "   
## 6 ( 1 ) " " "\*" "\*" " " " " " " " " " "   
## 7 ( 1 ) " " "\*" "\*" " " " " " " " " " "   
## 8 ( 1 ) " " "\*" "\*" " " " " " " " " " "   
## 9 ( 1 ) "\*" "\*" "\*" " " " " " " " " " "   
## 10 ( 1 ) "\*" "\*" "\*" " " " " " " " " "\*"   
## 11 ( 1 ) "\*" "\*" "\*" " " " " " " " " "\*"   
## 12 ( 1 ) "\*" "\*" "\*" " " " " " " " " "\*"   
## 13 ( 1 ) "\*" "\*" "\*" " " " " "\*" " " "\*"   
## 14 ( 1 ) "\*" "\*" "\*" "\*" " " "\*" " " "\*"   
## 15 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" " " "\*"   
## 16 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" " " "\*"   
## 17 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" " " "\*"   
## 18 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*"   
## 19 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*"   
## 20 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*"   
## IncomePerCap Poverty ChildPoverty Professional Service Office  
## 1 ( 1 ) " " " " " " " " " " " "   
## 2 ( 1 ) " " " " " " " " " " " "   
## 3 ( 1 ) "\*" " " " " " " " " " "   
## 4 ( 1 ) "\*" " " " " " " "\*" " "   
## 5 ( 1 ) "\*" " " " " " " "\*" " "   
## 6 ( 1 ) "\*" "\*" " " " " "\*" " "   
## 7 ( 1 ) "\*" "\*" " " " " "\*" " "   
## 8 ( 1 ) "\*" "\*" " " "\*" "\*" " "   
## 9 ( 1 ) "\*" "\*" " " "\*" "\*" " "   
## 10 ( 1 ) "\*" "\*" " " "\*" "\*" " "   
## 11 ( 1 ) "\*" "\*" " " "\*" "\*" " "   
## 12 ( 1 ) "\*" "\*" "\*" "\*" "\*" " "   
## 13 ( 1 ) "\*" "\*" "\*" "\*" "\*" " "   
## 14 ( 1 ) "\*" "\*" "\*" "\*" "\*" " "   
## 15 ( 1 ) "\*" "\*" "\*" "\*" "\*" " "   
## 16 ( 1 ) "\*" "\*" "\*" "\*" "\*" " "   
## 17 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*"   
## 18 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*"   
## 19 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*"   
## 20 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*"   
## Construction Production SelfEmployed FamilyWork Unemployment  
## 1 ( 1 ) " " " " " " " " " "   
## 2 ( 1 ) "\*" " " " " " " " "   
## 3 ( 1 ) "\*" " " " " " " " "   
## 4 ( 1 ) "\*" " " " " " " " "   
## 5 ( 1 ) "\*" " " " " " " " "   
## 6 ( 1 ) "\*" " " " " " " " "   
## 7 ( 1 ) "\*" " " "\*" " " " "   
## 8 ( 1 ) "\*" " " "\*" " " " "   
## 9 ( 1 ) "\*" " " "\*" " " " "   
## 10 ( 1 ) "\*" " " "\*" " " " "   
## 11 ( 1 ) "\*" " " "\*" " " "\*"   
## 12 ( 1 ) "\*" " " "\*" " " "\*"   
## 13 ( 1 ) "\*" " " "\*" " " "\*"   
## 14 ( 1 ) "\*" " " "\*" " " "\*"   
## 15 ( 1 ) "\*" " " "\*" " " "\*"   
## 16 ( 1 ) "\*" " " "\*" " " "\*"   
## 17 ( 1 ) "\*" " " "\*" " " "\*"   
## 18 ( 1 ) "\*" " " "\*" " " "\*"   
## 19 ( 1 ) "\*" "\*" "\*" " " "\*"   
## 20 ( 1 ) "\*" "\*" "\*" "\*" "\*"   
## RatioMenWomen  
## 1 ( 1 ) " "   
## 2 ( 1 ) " "   
## 3 ( 1 ) " "   
## 4 ( 1 ) " "   
## 5 ( 1 ) " "   
## 6 ( 1 ) " "   
## 7 ( 1 ) " "   
## 8 ( 1 ) " "   
## 9 ( 1 ) " "   
## 10 ( 1 ) " "   
## 11 ( 1 ) " "   
## 12 ( 1 ) " "   
## 13 ( 1 ) " "   
## 14 ( 1 ) " "   
## 15 ( 1 ) " "   
## 16 ( 1 ) "\*"   
## 17 ( 1 ) "\*"   
## 18 ( 1 ) "\*"   
## 19 ( 1 ) "\*"   
## 20 ( 1 ) "\*"

#plot(forward.model2020)  
  
fwdmod\_cp<-summary(forward.model2020)$cp  
fwdmod\_cp

## [1] 3476.89318 1307.30660 691.77965 450.80934 267.60179 195.10283  
## [7] 152.09657 122.28018 99.05314 84.54543 69.78659 57.23135  
## [13] 48.02917 39.07688 27.99728 19.20003 17.70514 18.35916  
## [19] 19.03317 21.00000

fwdmod\_bic<-summary(forward.model2020)$bic  
fwdmod\_bic

## [1] -918.1843 -2144.2439 -2602.8918 -2800.0551 -2958.2822 -3019.7794  
## [7] -3054.5374 -3077.1247 -3093.5932 -3101.7044 -3110.1489 -3116.5009  
## [13] -3119.5880 -3122.4667 -3127.5138 -3130.3229 -3125.8153 -3119.1481  
## [19] -3112.4614 -3104.4733

fwdmod\_adjr2<-summary(forward.model2020)$adjr2  
fwdmod\_adjr2

## [1] 0.2638916 0.5089442 0.5785592 0.6058837 0.6266951 0.6349985 0.6399704  
## [8] 0.6434529 0.6461915 0.6479445 0.6497271 0.6512612 0.6524161 0.6535435  
## [15] 0.6549129 0.6560242 0.6563073 0.6563466 0.6563837 0.6562739

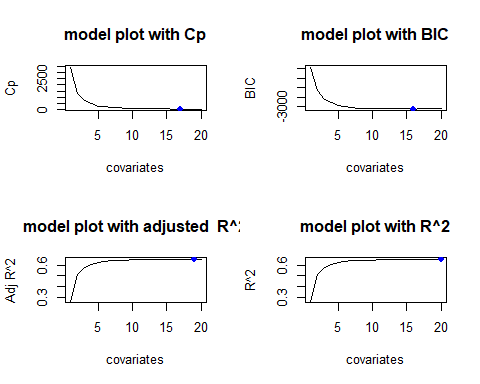
fwdmod\_r2<-summary(forward.model2020)$rsq  
fwdmod\_r2

## [1] 0.2641333 0.5092667 0.5789745 0.6064015 0.6273081 0.6357177 0.6407981  
## [8] 0.6443896 0.6472373 0.6491007 0.6509925 0.6526355 0.6539000 0.6551364  
## [15] 0.6566128 0.6578316 0.6582261 0.6583781 0.6585278 0.6585315

fwdmodel2020DF <- data.frame(numVars = 1:20, Cp = fwdmod\_cp,   
 BIC = fwdmod\_bic,  
 Adjr2 = fwdmod\_adjr2,  
 r2 = fwdmod\_r2)  
fwdmodel2020DF

## numVars Cp BIC Adjr2 r2  
## 1 1 3476.89318 -918.1843 0.2638916 0.2641333  
## 2 2 1307.30660 -2144.2439 0.5089442 0.5092667  
## 3 3 691.77965 -2602.8918 0.5785592 0.5789745  
## 4 4 450.80934 -2800.0551 0.6058837 0.6064015  
## 5 5 267.60179 -2958.2822 0.6266951 0.6273081  
## 6 6 195.10283 -3019.7794 0.6349985 0.6357177  
## 7 7 152.09657 -3054.5374 0.6399704 0.6407981  
## 8 8 122.28018 -3077.1247 0.6434529 0.6443896  
## 9 9 99.05314 -3093.5932 0.6461915 0.6472373  
## 10 10 84.54543 -3101.7044 0.6479445 0.6491007  
## 11 11 69.78659 -3110.1489 0.6497271 0.6509925  
## 12 12 57.23135 -3116.5009 0.6512612 0.6526355  
## 13 13 48.02917 -3119.5880 0.6524161 0.6539000  
## 14 14 39.07688 -3122.4667 0.6535435 0.6551364  
## 15 15 27.99728 -3127.5138 0.6549129 0.6566128  
## 16 16 19.20003 -3130.3229 0.6560242 0.6578316  
## 17 17 17.70514 -3125.8153 0.6563073 0.6582261  
## 18 18 18.35916 -3119.1481 0.6563466 0.6583781  
## 19 19 19.03317 -3112.4614 0.6563837 0.6585278  
## 20 20 21.00000 -3104.4733 0.6562739 0.6585315

par(mfrow=c(2,2))  
  
plot(fwdmod\_cp,main= "model plot with Cp",xlab="covariates",ylab="Cp",type="l")  
points(which.min(fwdmod\_cp),fwdmod\_cp[which.min(fwdmod\_cp)],col="blue",cex=1,pch=16)  
  
plot(fwdmod\_bic,main = "model plot with BIC",xlab="covariates",ylab="BIC",type="l")  
points(which.min(fwdmod\_bic),fwdmod\_bic[which.min(fwdmod\_bic)],col="blue",cex=1,pch=16)  
  
plot(fwdmod\_adjr2,main = "model plot with adjusted R^2",xlab="covariates",ylab="Adj R^2",type="l")  
points(which.max(fwdmod\_adjr2),fwdmod\_adjr2[which.max(fwdmod\_adjr2)],col="blue",cex=1,pch=16)  
  
plot(fwdmod\_r2,main = "model plot with R^2",xlab="covariates",ylab="R^2",type="l")  
points(which.max(fwdmod\_r2),fwdmod\_r2[which.max(fwdmod\_r2)],col="blue",cex=1,pch=16)



cp\_coef<-coef(forward.model2020,which.min(fwdmod\_cp))  
cp\_coef

## (Intercept) TotalPop Hispanic White Black   
## 1.017713e+00 -1.944200e-08 -2.546025e-03 8.594081e-05 -4.452884e-03   
## Native Asian Income IncomePerCap Poverty   
## -3.971285e-03 -8.606069e-03 1.339599e-06 -1.168057e-05 -3.692913e-03   
## ChildPoverty Professional Service Office Construction   
## 1.655861e-03 -2.224922e-03 -8.497764e-03 1.195244e-03 8.479091e-03   
## SelfEmployed Unemployment RatioMenWomen   
## 4.294195e-03 -3.405665e-03 4.962868e-02

bic\_coef<-coef(forward.model2020,which.min(fwdmod\_bic))  
bic\_coef

## (Intercept) TotalPop Hispanic White Black   
## 1.078210e+00 -1.843305e-08 -2.749596e-03 -1.414294e-04 -4.674715e-03   
## Native Asian Income IncomePerCap Poverty   
## -4.217578e-03 -9.000404e-03 1.346625e-06 -1.162074e-05 -3.764170e-03   
## ChildPoverty Professional Service Construction SelfEmployed   
## 1.678593e-03 -2.355968e-03 -8.638157e-03 8.120136e-03 4.167384e-03   
## Unemployment RatioMenWomen   
## -3.283497e-03 4.749373e-02

adjr2\_coef<-coef(forward.model2020,which.max(fwdmod\_adjr2))  
adjr2\_coef

## (Intercept) TotalPop Hispanic White Black   
## -1.836700e+00 -2.037520e-08 -3.222159e-03 -5.890689e-04 -5.120194e-03   
## Native Asian Pacific Income IncomePerCap   
## -4.667064e-03 -8.825000e-03 -6.647558e-03 1.323755e-06 -1.170075e-05   
## Poverty ChildPoverty Professional Service Office   
## -3.711020e-03 1.641686e-03 2.695335e-02 2.074771e-02 3.043625e-02   
## Construction Production SelfEmployed Unemployment RatioMenWomen   
## 3.774476e-02 2.923312e-02 4.336866e-03 -3.449832e-03 4.958602e-02

r2\_coef<-coef(forward.model2020,which.max(fwdmod\_r2))  
r2\_coef

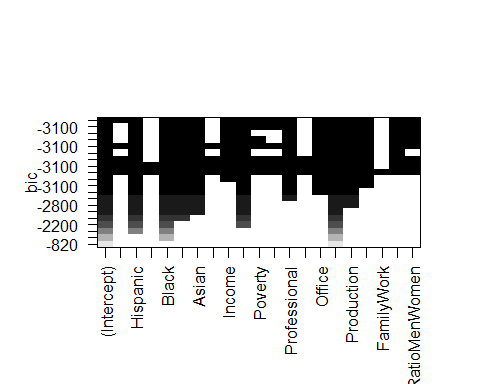
## (Intercept) TotalPop Hispanic White Black   
## -1.846921e+00 -2.036460e-08 -3.224345e-03 -5.899842e-04 -5.121689e-03   
## Native Asian Pacific Income IncomePerCap   
## -4.664813e-03 -8.825566e-03 -6.646815e-03 1.320777e-06 -1.169903e-05   
## Poverty ChildPoverty Professional Service Office   
## -3.707242e-03 1.639823e-03 2.705983e-02 2.084868e-02 3.053541e-02   
## Construction Production SelfEmployed FamilyWork Unemployment   
## 3.785734e-02 2.933447e-02 4.357444e-03 -7.231303e-04 -3.453703e-03   
## RatioMenWomen   
## 4.967528e-02

# backward 2020

backward.model2020<-regsubsets(ptrump20 ~ ., data = Trump20Reg, method ="backward", nvmax = 21)  
summary(backward.model2020)

## Subset selection object  
## Call: regsubsets.formula(ptrump20 ~ ., data = Trump20Reg, method = "backward",   
## nvmax = 21)  
## 20 Variables (and intercept)  
## Forced in Forced out  
## TotalPop FALSE FALSE  
## Hispanic FALSE FALSE  
## White FALSE FALSE  
## Black FALSE FALSE  
## Native FALSE FALSE  
## Asian FALSE FALSE  
## Pacific FALSE FALSE  
## Income FALSE FALSE  
## IncomePerCap FALSE FALSE  
## Poverty FALSE FALSE  
## ChildPoverty FALSE FALSE  
## Professional FALSE FALSE  
## Service FALSE FALSE  
## Office FALSE FALSE  
## Construction FALSE FALSE  
## Production FALSE FALSE  
## SelfEmployed FALSE FALSE  
## FamilyWork FALSE FALSE  
## Unemployment FALSE FALSE  
## RatioMenWomen FALSE FALSE  
## 1 subsets of each size up to 20  
## Selection Algorithm: backward  
## TotalPop Hispanic White Black Native Asian Pacific Income  
## 1 ( 1 ) " " " " " " " " " " " " " " " "   
## 2 ( 1 ) " " " " " " "\*" " " " " " " " "   
## 3 ( 1 ) " " "\*" " " "\*" " " " " " " " "   
## 4 ( 1 ) " " "\*" " " "\*" " " " " " " " "   
## 5 ( 1 ) " " "\*" " " "\*" "\*" " " " " " "   
## 6 ( 1 ) " " "\*" " " "\*" "\*" "\*" " " " "   
## 7 ( 1 ) " " "\*" " " "\*" "\*" "\*" " " " "   
## 8 ( 1 ) " " "\*" " " "\*" "\*" "\*" " " " "   
## 9 ( 1 ) " " "\*" " " "\*" "\*" "\*" " " " "   
## 10 ( 1 ) " " "\*" " " "\*" "\*" "\*" " " " "   
## 11 ( 1 ) " " "\*" " " "\*" "\*" "\*" " " "\*"   
## 12 ( 1 ) " " "\*" " " "\*" "\*" "\*" " " "\*"   
## 13 ( 1 ) " " "\*" " " "\*" "\*" "\*" " " "\*"   
## 14 ( 1 ) " " "\*" " " "\*" "\*" "\*" " " "\*"   
## 15 ( 1 ) " " "\*" " " "\*" "\*" "\*" " " "\*"   
## 16 ( 1 ) "\*" "\*" " " "\*" "\*" "\*" " " "\*"   
## 17 ( 1 ) "\*" "\*" " " "\*" "\*" "\*" "\*" "\*"   
## 18 ( 1 ) "\*" "\*" " " "\*" "\*" "\*" "\*" "\*"   
## 19 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*"   
## 20 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*"   
## IncomePerCap Poverty ChildPoverty Professional Service Office  
## 1 ( 1 ) " " " " " " " " " " " "   
## 2 ( 1 ) " " " " " " " " " " " "   
## 3 ( 1 ) " " " " " " " " " " " "   
## 4 ( 1 ) "\*" " " " " " " " " " "   
## 5 ( 1 ) "\*" " " " " " " " " " "   
## 6 ( 1 ) "\*" " " " " " " " " " "   
## 7 ( 1 ) "\*" " " " " " " " " " "   
## 8 ( 1 ) "\*" " " " " "\*" " " " "   
## 9 ( 1 ) "\*" " " " " "\*" " " "\*"   
## 10 ( 1 ) "\*" " " " " "\*" " " "\*"   
## 11 ( 1 ) "\*" " " " " "\*" " " "\*"   
## 12 ( 1 ) "\*" " " " " "\*" " " "\*"   
## 13 ( 1 ) "\*" " " " " "\*" " " "\*"   
## 14 ( 1 ) "\*" "\*" " " "\*" " " "\*"   
## 15 ( 1 ) "\*" "\*" "\*" "\*" " " "\*"   
## 16 ( 1 ) "\*" "\*" "\*" "\*" " " "\*"   
## 17 ( 1 ) "\*" "\*" "\*" "\*" " " "\*"   
## 18 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*"   
## 19 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*"   
## 20 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*"   
## Construction Production SelfEmployed FamilyWork Unemployment  
## 1 ( 1 ) "\*" " " " " " " " "   
## 2 ( 1 ) "\*" " " " " " " " "   
## 3 ( 1 ) "\*" " " " " " " " "   
## 4 ( 1 ) "\*" " " " " " " " "   
## 5 ( 1 ) "\*" " " " " " " " "   
## 6 ( 1 ) "\*" " " " " " " " "   
## 7 ( 1 ) "\*" "\*" " " " " " "   
## 8 ( 1 ) "\*" "\*" " " " " " "   
## 9 ( 1 ) "\*" "\*" " " " " " "   
## 10 ( 1 ) "\*" "\*" "\*" " " " "   
## 11 ( 1 ) "\*" "\*" "\*" " " " "   
## 12 ( 1 ) "\*" "\*" "\*" " " "\*"   
## 13 ( 1 ) "\*" "\*" "\*" " " "\*"   
## 14 ( 1 ) "\*" "\*" "\*" " " "\*"   
## 15 ( 1 ) "\*" "\*" "\*" " " "\*"   
## 16 ( 1 ) "\*" "\*" "\*" " " "\*"   
## 17 ( 1 ) "\*" "\*" "\*" " " "\*"   
## 18 ( 1 ) "\*" "\*" "\*" " " "\*"   
## 19 ( 1 ) "\*" "\*" "\*" " " "\*"   
## 20 ( 1 ) "\*" "\*" "\*" "\*" "\*"   
## RatioMenWomen  
## 1 ( 1 ) " "   
## 2 ( 1 ) " "   
## 3 ( 1 ) " "   
## 4 ( 1 ) " "   
## 5 ( 1 ) " "   
## 6 ( 1 ) " "   
## 7 ( 1 ) " "   
## 8 ( 1 ) " "   
## 9 ( 1 ) " "   
## 10 ( 1 ) " "   
## 11 ( 1 ) " "   
## 12 ( 1 ) " "   
## 13 ( 1 ) "\*"   
## 14 ( 1 ) "\*"   
## 15 ( 1 ) "\*"   
## 16 ( 1 ) "\*"   
## 17 ( 1 ) "\*"   
## 18 ( 1 ) "\*"   
## 19 ( 1 ) "\*"   
## 20 ( 1 ) "\*"

plot(backward.model2020)



bwdmod\_cp<-summary(backward.model2020)$cp  
bwdmod\_cp

## [1] 3681.55237 2632.59005 1807.74950 1151.58593 715.50665 501.96746  
## [7] 395.43017 338.46853 189.57626 136.93917 79.48816 49.75218  
## [13] 36.74726 33.01880 23.40778 15.04657 15.94391 17.27335  
## [19] 19.03317 21.00000

bwdmod\_bic<-summary(backward.model2020)$bic  
bwdmod\_bic

## [1] -824.0263 -1333.7358 -1805.5929 -2242.2188 -2570.7807 -2743.0918  
## [7] -2830.0874 -2874.9529 -3006.6138 -3050.7927 -3100.6060 -3123.9132  
## [13] -3130.8169 -3128.5127 -3132.1128 -3134.4995 -3127.5877 -3120.2412  
## [19] -3112.4614 -3104.4733

bwdmod\_adjr2<-summary(backward.model2020)$adjr2  
bwdmod\_adjr2

## [1] 0.2407816 0.3592450 0.4524619 0.5266745 0.5760515 0.6002906 0.6124391  
## [8] 0.6189848 0.6359428 0.6420107 0.6486280 0.6521088 0.6536951 0.6542305  
## [15] 0.6554335 0.6564955 0.6565073 0.6564699 0.6563837 0.6562739

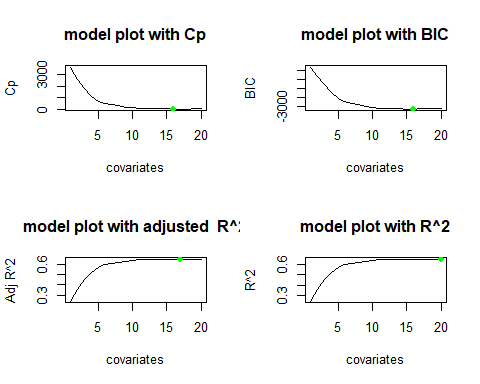
bwdmod\_r2<-summary(backward.model2020)$rsq  
bwdmod\_r2

## [1] 0.2410310 0.3596658 0.4530014 0.5272963 0.5767476 0.6010782 0.6133301  
## [8] 0.6199858 0.6370188 0.6431864 0.6498974 0.6534798 0.6551736 0.6558202  
## [15] 0.6571309 0.6583005 0.6584250 0.6585006 0.6585278 0.6585315

bwdmodel2020DF <- data.frame(numVars = 1:20, Cp = bwdmod\_cp,   
 BIC = bwdmod\_bic,  
 Adjr2 = bwdmod\_adjr2,  
 r2 = bwdmod\_r2)  
bwdmodel2020DF

## numVars Cp BIC Adjr2 r2  
## 1 1 3681.55237 -824.0263 0.2407816 0.2410310  
## 2 2 2632.59005 -1333.7358 0.3592450 0.3596658  
## 3 3 1807.74950 -1805.5929 0.4524619 0.4530014  
## 4 4 1151.58593 -2242.2188 0.5266745 0.5272963  
## 5 5 715.50665 -2570.7807 0.5760515 0.5767476  
## 6 6 501.96746 -2743.0918 0.6002906 0.6010782  
## 7 7 395.43017 -2830.0874 0.6124391 0.6133301  
## 8 8 338.46853 -2874.9529 0.6189848 0.6199858  
## 9 9 189.57626 -3006.6138 0.6359428 0.6370188  
## 10 10 136.93917 -3050.7927 0.6420107 0.6431864  
## 11 11 79.48816 -3100.6060 0.6486280 0.6498974  
## 12 12 49.75218 -3123.9132 0.6521088 0.6534798  
## 13 13 36.74726 -3130.8169 0.6536951 0.6551736  
## 14 14 33.01880 -3128.5127 0.6542305 0.6558202  
## 15 15 23.40778 -3132.1128 0.6554335 0.6571309  
## 16 16 15.04657 -3134.4995 0.6564955 0.6583005  
## 17 17 15.94391 -3127.5877 0.6565073 0.6584250  
## 18 18 17.27335 -3120.2412 0.6564699 0.6585006  
## 19 19 19.03317 -3112.4614 0.6563837 0.6585278  
## 20 20 21.00000 -3104.4733 0.6562739 0.6585315

par(mfrow=c(2,2))  
  
plot(bwdmod\_cp,main= "model plot with Cp",xlab="covariates",ylab="Cp",type="l")  
points(which.min(bwdmod\_cp),bwdmod\_cp[which.min(bwdmod\_cp)],col="green",cex=1,pch=16)  
  
plot(bwdmod\_bic,main = "model plot with BIC",xlab="covariates",ylab="BIC",type="l")  
points(which.min(bwdmod\_bic),bwdmod\_bic[which.min(bwdmod\_bic)],col="green",cex=1,pch=16)  
  
plot(bwdmod\_adjr2,main = "model plot with adjusted R^2",xlab="covariates",ylab="Adj R^2",type="l")  
points(which.max(bwdmod\_adjr2),bwdmod\_adjr2[which.max(bwdmod\_adjr2)],col="green",cex=1,pch=16)  
  
plot(bwdmod\_r2,main = "model plot with R^2",xlab="covariates",ylab="R^2",type="l")  
points(which.max(bwdmod\_r2),bwdmod\_r2[which.max(bwdmod\_r2)],col="green",cex=1,pch=16)



cp\_coef<-coef(backward.model2020,which.min(bwdmod\_cp))  
bic\_coef<-coef(backward.model2020,which.min(bwdmod\_bic))  
adjr2\_coef<-coef(backward.model2020,which.max(bwdmod\_adjr2))  
r2\_coef<-coef(backward.model2020,which.max(bwdmod\_r2))  
cp\_coef

## (Intercept) TotalPop Hispanic Black Native   
## 1.748871e-01 -1.939720e-08 -2.629746e-03 -4.536607e-03 -4.061915e-03   
## Asian Income IncomePerCap Poverty ChildPoverty   
## -8.722826e-03 1.335012e-06 -1.166245e-05 -3.686704e-03 1.653781e-03   
## Professional Office Construction Production SelfEmployed   
## 6.286753e-03 9.701021e-03 1.698839e-02 8.517806e-03 4.291127e-03   
## Unemployment RatioMenWomen   
## -3.407008e-03 4.974542e-02

bic\_coef

## (Intercept) TotalPop Hispanic Black Native   
## 1.748871e-01 -1.939720e-08 -2.629746e-03 -4.536607e-03 -4.061915e-03   
## Asian Income IncomePerCap Poverty ChildPoverty   
## -8.722826e-03 1.335012e-06 -1.166245e-05 -3.686704e-03 1.653781e-03   
## Professional Office Construction Production SelfEmployed   
## 6.286753e-03 9.701021e-03 1.698839e-02 8.517806e-03 4.291127e-03   
## Unemployment RatioMenWomen   
## -3.407008e-03 4.974542e-02

adjr2\_coef

## (Intercept) TotalPop Hispanic Black Native   
## 1.814060e-01 -2.040199e-08 -2.640261e-03 -4.540032e-03 -4.045842e-03   
## Asian Pacific Income IncomePerCap Poverty   
## -8.160812e-03 -5.297454e-03 1.331453e-06 -1.173858e-05 -3.729657e-03   
## ChildPoverty Professional Office Construction Production   
## 1.649357e-03 6.203132e-03 9.701727e-03 1.698281e-02 8.461818e-03   
## SelfEmployed Unemployment RatioMenWomen   
## 4.332849e-03 -3.431639e-03 4.946153e-02

r2\_coef

## (Intercept) TotalPop Hispanic White Black   
## -1.846921e+00 -2.036460e-08 -3.224345e-03 -5.899842e-04 -5.121689e-03   
## Native Asian Pacific Income IncomePerCap   
## -4.664813e-03 -8.825566e-03 -6.646815e-03 1.320777e-06 -1.169903e-05   
## Poverty ChildPoverty Professional Service Office   
## -3.707242e-03 1.639823e-03 2.705983e-02 2.084868e-02 3.053541e-02   
## Construction Production SelfEmployed FamilyWork Unemployment   
## 3.785734e-02 2.933447e-02 4.357444e-03 -7.231303e-04 -3.453703e-03   
## RatioMenWomen   
## 4.967528e-02

# SVM MODELS

library(caret)  
# install.packages('e1071')  
library(e1071)

# 2016 SVM

# Splitting the dataset into the Training set and Test set

set.seed(1234)  
split = sample.split(Trump16Class$TrumpOrClinton, SplitRatio = 0.80)  
training16 = subset(Trump16Class, split == TRUE)  
test16 = subset(Trump16Class, split == FALSE)  
#names(training16)  
# Feature Scaling  
training16[-21] = scale(training16[-21])  
test16[-21] = scale(test16[-21])

# Fitting SVM to the Training set  
svm.model16 = svm(formula = TrumpOrClinton ~.,  
 data = training16,  
 type = 'C-classification',  
 kernel = 'linear')  
  
summary(svm.model16)

##   
## Call:  
## svm(formula = TrumpOrClinton ~ ., data = training16, type = "C-classification",   
## kernel = "linear")  
##   
##   
## Parameters:  
## SVM-Type: C-classification   
## SVM-Kernel: linear   
## cost: 1   
##   
## Number of Support Vectors: 479  
##   
## ( 241 238 )  
##   
##   
## Number of Classes: 2   
##   
## Levels:   
## Clinton Trump

#svm.model16$index  
#print(svm.model16$best.parameters)

# Predicting the Test set results  
svm.pred16 = predict(svm.model16, newdata = test16[-21])  
  
# Making the Confusion Matrix  
confmat16 = table(test16[, 21], svm.pred16)  
confmat16

## svm.pred16  
## Clinton Trump  
## Clinton 59 30  
## Trump 13 507

## test error  
mean(svm.pred16 != test16$TrumpOrClinton)

## [1] 0.07060755

## test accuracy  
mean(svm.pred16 == test16$TrumpOrClinton)

## [1] 0.9293924

#plot(training16, svm.pred16)

# tuning2016

set.seed(1234)  
svm1.model16 = svm(formula = TrumpOrClinton ~.,  
 data = training16,  
 type = 'C-classification',  
 kernel = 'linear',cost = 10, scale = FALSE)  
  
summary(svm1.model16)

##   
## Call:  
## svm(formula = TrumpOrClinton ~ ., data = training16, type = "C-classification",   
## kernel = "linear", cost = 10, scale = FALSE)  
##   
##   
## Parameters:  
## SVM-Type: C-classification   
## SVM-Kernel: linear   
## cost: 10   
##   
## Number of Support Vectors: 478  
##   
## ( 242 236 )  
##   
##   
## Number of Classes: 2   
##   
## Levels:   
## Clinton Trump

svm1.model16$index

## [1] 19 23 27 77 84 91 107 111 117 121 126 127 135 140 142  
## [16] 152 156 161 189 197 210 211 214 215 240 244 256 263 267 284  
## [31] 324 366 378 386 399 409 447 459 465 473 474 475 483 487 501  
## [46] 506 523 530 553 554 560 573 585 590 592 597 604 619 621 622  
## [61] 636 638 639 645 650 662 663 679 727 728 753 766 775 794 808  
## [76] 814 828 843 858 863 865 870 872 876 895 897 902 916 924 935  
## [91] 946 973 1012 1021 1035 1046 1067 1073 1082 1084 1104 1105 1120 1141 1149  
## [106] 1152 1156 1162 1167 1181 1195 1198 1214 1218 1226 1231 1234 1249 1251 1253  
## [121] 1267 1302 1332 1333 1334 1337 1347 1350 1356 1361 1365 1371 1378 1393 1408  
## [136] 1412 1441 1444 1446 1456 1487 1497 1507 1511 1515 1518 1519 1520 1543 1546  
## [151] 1552 1554 1568 1569 1588 1592 1594 1596 1609 1643 1649 1685 1694 1695 1702  
## [166] 1708 1723 1724 1728 1751 1766 1773 1778 1783 1784 1789 1794 1816 1831 1836  
## [181] 1849 1853 1862 1870 1874 1890 1907 1917 1921 1922 1942 1960 1971 1975 1997  
## [196] 2004 2005 2006 2014 2044 2057 2063 2082 2085 2089 2091 2092 2095 2096 2125  
## [211] 2146 2147 2153 2159 2168 2175 2179 2186 2193 2208 2210 2211 2213 2244 2261  
## [226] 2271 2274 2288 2299 2321 2329 2338 2348 2350 2353 2377 2386 2387 2409 2432  
## [241] 2434 2435 8 21 22 24 34 39 40 53 55 58 63 73 80  
## [256] 97 100 110 124 141 157 162 167 169 191 217 223 234 246 248  
## [271] 270 279 294 311 348 355 357 365 382 395 403 406 412 426 432  
## [286] 435 453 462 479 495 500 505 524 531 535 536 537 556 561 563  
## [301] 579 588 600 603 609 624 626 628 631 640 641 648 681 691 704  
## [316] 740 746 751 760 771 773 802 803 821 829 835 853 856 869 883  
## [331] 888 889 901 905 948 959 968 974 986 992 1015 1032 1034 1047 1078  
## [346] 1079 1083 1085 1091 1100 1103 1113 1117 1134 1138 1146 1169 1177 1185 1187  
## [361] 1201 1203 1205 1220 1224 1228 1252 1259 1264 1280 1289 1303 1335 1349 1372  
## [376] 1377 1386 1391 1394 1395 1403 1466 1468 1478 1516 1523 1527 1528 1532 1544  
## [391] 1547 1555 1584 1590 1607 1610 1616 1618 1636 1641 1666 1673 1674 1680 1716  
## [406] 1717 1736 1754 1764 1776 1796 1808 1827 1830 1832 1833 1834 1856 1873 1910  
## [421] 1911 1914 1916 1919 1926 1932 1937 1943 1948 1952 1958 1961 1963 1969 1981  
## [436] 1989 1991 2045 2052 2059 2061 2066 2074 2078 2080 2081 2083 2087 2121 2122  
## [451] 2133 2134 2135 2139 2140 2141 2145 2152 2154 2170 2180 2192 2222 2245 2256  
## [466] 2269 2283 2289 2307 2310 2317 2336 2357 2381 2384 2392 2415 2437

#print(svm1.model16$best.parameters)

# Predicting the Test set results  
svm1.pred16 = predict(svm1.model16, newdata = test16[-21])  
  
# Making the Confusion Matrix  
confmat1.16 = table(test16[, 21], svm1.pred16)  
confmat1.16

## svm1.pred16  
## Clinton Trump  
## Clinton 58 31  
## Trump 12 508

## test error  
mean(svm1.pred16 != test16$TrumpOrClinton)

## [1] 0.07060755

## test accuracy  
mean(svm1.pred16 == test16$TrumpOrClinton)

## [1] 0.9293924

#plot(training16, svm.pred16)

## can tune multiple functions

set.seed(1234)  
  
tuneModel16 <- tune(svm, TrumpOrClinton ~., data = training16,  
 type = 'C-classification',  
 kernel = "linear",  
 ranges = list(cost = c(0.001, 0.01, 0.1,  
 1, 5, 10, 100, 1000)))  
  
# Predicting the Test set results  
  
bestSVM16 <- tuneModel16$best.model  
  
bestSVM16

##   
## Call:  
## best.tune(method = svm, train.x = TrumpOrClinton ~ ., data = training16,   
## ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5, 10, 100, 1000)),   
## type = "C-classification", kernel = "linear")  
##   
##   
## Parameters:  
## SVM-Type: C-classification   
## SVM-Kernel: linear   
## cost: 1   
##   
## Number of Support Vectors: 479

svmTune.pred16 = predict(bestSVM16, newdata = test16[-21])  
  
# Making the Confusion Matrix  
confmattune.16 = table(test16[, 21], svmTune.pred16)  
confmattune.16

## svmTune.pred16  
## Clinton Trump  
## Clinton 59 30  
## Trump 13 507

## test error  
mean(svmTune.pred16 != test16$TrumpOrClinton)

## [1] 0.07060755

## test accuracy  
mean(svmTune.pred16 == test16$TrumpOrClinton)

## [1] 0.9293924

# 2020 SVM

set.seed(1234)  
split = sample.split(Trump20Class$TrumpOrBiden, SplitRatio = 0.80)  
training20 = subset(Trump20Class, split == TRUE)  
test20 = subset(Trump20Class, split == FALSE)  
#names(training20)  
# Feature Scaling  
training20[-21] = scale(training20[-21])  
test20 [-21] = scale(test20 [-21])

# Fitting SVM to the Training set  
svm.model20 = svm(formula = TrumpOrBiden ~.,  
 data = training20,  
 type = 'C-classification',  
 kernel = 'linear')  
summary(svm.model20)

##   
## Call:  
## svm(formula = TrumpOrBiden ~ ., data = training20, type = "C-classification",   
## kernel = "linear")  
##   
##   
## Parameters:  
## SVM-Type: C-classification   
## SVM-Kernel: linear   
## cost: 1   
##   
## Number of Support Vectors: 490  
##   
## ( 248 242 )  
##   
##   
## Number of Classes: 2   
##   
## Levels:   
## Biden Trump

svm.model20$index

## [1] 4 21 27 35 62 82 89 108 118 123 124 137 138 147 150  
## [16] 154 182 204 208 252 257 263 268 280 302 320 333 363 375 397  
## [31] 399 405 408 447 448 459 473 474 475 476 480 485 487 490 499  
## [46] 500 514 522 557 586 589 592 600 605 615 617 618 620 634 642  
## [61] 659 660 678 694 696 698 715 725 762 774 787 806 809 818 824  
## [76] 838 853 861 866 868 871 890 892 897 922 931 970 976 996 998  
## [91] 1009 1018 1032 1044 1054 1066 1085 1103 1104 1144 1162 1165 1170 1172 1177  
## [106] 1186 1196 1198 1202 1213 1218 1226 1230 1249 1251 1252 1264 1292 1299 1327  
## [121] 1328 1329 1330 1335 1343 1344 1351 1356 1361 1362 1368 1370 1392 1403 1407  
## [136] 1423 1428 1438 1441 1451 1457 1483 1494 1503 1508 1513 1516 1517 1518 1519  
## [151] 1540 1542 1548 1551 1571 1583 1589 1590 1635 1640 1647 1655 1684 1692 1701  
## [166] 1709 1715 1722 1723 1726 1750 1763 1780 1782 1786 1793 1816 1829 1835 1845  
## [181] 1850 1858 1869 1872 1890 1907 1912 1913 1918 1919 1956 1957 1968 1975 1976  
## [196] 1991 2004 2005 2019 2050 2056 2067 2068 2081 2087 2088 2089 2090 2110 2124  
## [211] 2142 2143 2144 2150 2161 2164 2171 2177 2194 2207 2210 2211 2213 2229 2231  
## [226] 2241 2242 2258 2262 2273 2275 2288 2297 2322 2344 2346 2348 2370 2373 2383  
## [241] 2390 2395 2405 2409 2430 2432 2433 2435 22 23 24 39 40 53 55  
## [256] 58 97 107 121 155 160 162 163 173 184 210 216 217 229 235  
## [271] 240 241 248 259 267 275 308 345 347 351 352 362 379 388 393  
## [286] 401 406 418 422 424 430 431 433 439 462 481 504 523 529 534  
## [301] 535 549 558 574 584 595 596 599 606 616 621 623 626 629 639  
## [316] 645 650 677 690 702 726 739 767 772 776 800 805 823 825 833  
## [331] 849 851 877 882 885 901 902 944 945 957 965 983 989 1013 1027  
## [346] 1030 1034 1038 1046 1047 1057 1062 1072 1078 1079 1084 1090 1100 1134 1137  
## [361] 1143 1147 1148 1169 1183 1185 1193 1200 1201 1204 1206 1224 1228 1258 1261  
## [376] 1278 1300 1332 1346 1374 1388 1389 1393 1394 1420 1460 1464 1475 1502 1514  
## [391] 1522 1526 1531 1550 1585 1587 1601 1603 1606 1611 1621 1631 1632 1633 1639  
## [406] 1665 1681 1693 1716 1731 1753 1770 1774 1788 1795 1808 1824 1828 1831 1833  
## [421] 1892 1908 1910 1916 1924 1935 1940 1941 1946 1949 1955 1958 1959 1961 1962  
## [436] 1966 1973 1978 1981 1992 2042 2046 2052 2054 2060 2072 2076 2078 2079 2080  
## [451] 2082 2085 2092 2093 2112 2121 2127 2131 2132 2133 2137 2138 2139 2140 2149  
## [466] 2151 2157 2169 2172 2180 2190 2222 2250 2285 2286 2299 2307 2316 2320 2321  
## [481] 2352 2377 2378 2379 2384 2387 2393 2416 2436 2437

# Predicting the Test set results  
svm.pred20 = predict(svm.model20, newdata = test20 [-21])  
# Making the Confusion Matrix  
confmat20 = table(test20 [, 21], svm.pred20)  
confmat20

## svm.pred20  
## Biden Trump  
## Biden 54 40  
## Trump 12 503

## test error  
mean(svm.pred20 != test20 $TrumpOrBiden)

## [1] 0.08538588

## test accuracy  
mean(svm.pred20 == test20 $TrumpOrBiden)

## [1] 0.9146141

#plot(training20, svm.pred20)

# tuning 2020

set.seed(1234)  
svm1.model20 = svm(formula = TrumpOrBiden ~.,  
 data = training20,  
 type = 'C-classification',  
 kernel = 'linear',cost = 10, scale = FALSE)  
  
#summary(svm1.model20)  
#svm1.model20$index  
#print(svm1.model20$best.parameters)

# Predicting the Test set results  
svm1.pred20 = predict(svm1.model20, newdata = test20 [-21])  
  
# Making the Confusion Matrix  
confmat1.20 = table(test20 [, 21], svm1.pred20)  
confmat1.20

## svm1.pred20  
## Biden Trump  
## Biden 55 39  
## Trump 13 502

## test error  
mean(svm1.pred20 != test20$TrumpOrBiden)

## [1] 0.08538588

## test accuracy  
mean(svm1.pred20 == test20$TrumpOrBiden)

## [1] 0.9146141

#plot(training20, svm1.pred20)

## tune multiple functions

set.seed(1234)  
tuneModel20 <- tune(svm, TrumpOrBiden ~., data = training20,  
 type = 'C-classification',  
 kernel = "linear",  
 ranges = list(cost = c(0.001, 0.01, 0.1,  
 1, 5, 10, 100, 1000)))

# Predicting the Test set results  
bestSVM20 <- tuneModel20$best.model  
bestSVM20

##   
## Call:  
## best.tune(method = svm, train.x = TrumpOrBiden ~ ., data = training20,   
## ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5, 10, 100, 1000)),   
## type = "C-classification", kernel = "linear")  
##   
##   
## Parameters:  
## SVM-Type: C-classification   
## SVM-Kernel: linear   
## cost: 0.1   
##   
## Number of Support Vectors: 510

svmTune.pred20 = predict(bestSVM20, newdata = test20 [-21])  
  
# Making the Confusion Matrix  
confmattune.20 = table(test20 [, 21], svmTune.pred20)  
confmattune.20

## svmTune.pred20  
## Biden Trump  
## Biden 56 38  
## Trump 12 503

## test error  
mean(svmTune.pred20 != test20 $TrumpOrBiden)

## [1] 0.08210181

## test accuracy  
mean(svmTune.pred20 == test20 $TrumpOrBiden)

## [1] 0.9178982

# Tree models

library(tree)

## Warning: package 'tree' was built under R version 4.0.5

## Registered S3 method overwritten by 'tree':  
## method from  
## print.tree cli

library(randomForest)

## Warning: package 'randomForest' was built under R version 4.0.5

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##   
## Attaching package: 'randomForest'

## The following object is masked from 'package:gridExtra':  
##   
## combine

## The following object is masked from 'package:dplyr':  
##   
## combine

## The following object is masked from 'package:ggplot2':  
##   
## margin

library(gbm)

## Warning: package 'gbm' was built under R version 4.0.5

## Loaded gbm 2.1.8

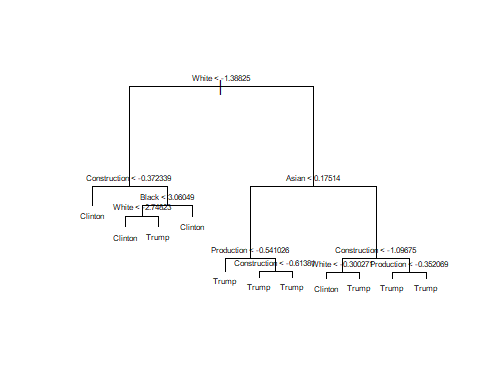
# 2016

set.seed(1234)  
split = sample.split(Trump16Class$TrumpOrClinton, SplitRatio = 0.80)  
training16 = subset(Trump16Class, split == TRUE)  
test16 = subset(Trump16Class, split == FALSE)  
#names(training16)  
# Feature Scaling  
training16[-21] = scale(training16[-21])  
test16[-21] = scale(test16[-21])

treeMod16 <- tree(TrumpOrClinton ~ ., data = training16)  
summary(treeMod16 )

##   
## Classification tree:  
## tree(formula = TrumpOrClinton ~ ., data = training16)  
## Variables actually used in tree construction:  
## [1] "White" "Construction" "Black" "Asian" "Production"   
## Number of terminal nodes: 11   
## Residual mean deviance: 0.3805 = 923.1 / 2426   
## Misclassification error rate: 0.07181 = 175 / 2437

plot(treeMod16 )  
## Can't read the labels very well  
text(treeMod16 , pretty = 0, cex = 0.5)



treeMod16

## node), split, n, deviance, yval, (yprob)  
## \* denotes terminal node  
##   
## 1) root 2437 2023.000 Trump ( 0.145671 0.854329 )   
## 2) White < -1.38825 270 351.400 Clinton ( 0.644444 0.355556 )   
## 4) Construction < -0.372339 113 52.500 Clinton ( 0.938053 0.061947 ) \*  
## 5) Construction > -0.372339 157 214.800 Trump ( 0.433121 0.566879 )   
## 10) Black < 3.06049 134 171.000 Trump ( 0.335821 0.664179 )   
## 20) White < -2.74823 22 8.136 Clinton ( 0.954545 0.045455 ) \*  
## 21) White > -2.74823 112 116.400 Trump ( 0.214286 0.785714 ) \*  
## 11) Black > 3.06049 23 0.000 Clinton ( 1.000000 0.000000 ) \*  
## 3) White > -1.38825 2167 1245.000 Trump ( 0.083526 0.916474 )   
## 6) Asian < 0.17514 1815 472.000 Trump ( 0.028650 0.971350 )   
## 12) Production < -0.541026 447 260.000 Trump ( 0.085011 0.914989 ) \*  
## 13) Production > -0.541026 1368 156.200 Trump ( 0.010234 0.989766 )   
## 26) Construction < -0.61381 245 89.770 Trump ( 0.044898 0.955102 ) \*  
## 27) Construction > -0.61381 1123 41.540 Trump ( 0.002671 0.997329 ) \*  
## 7) Asian > 0.17514 352 462.600 Trump ( 0.366477 0.633523 )   
## 14) Construction < -1.09675 165 224.900 Clinton ( 0.575758 0.424242 )   
## 28) White < -0.300271 60 57.170 Clinton ( 0.816667 0.183333 ) \*  
## 29) White > -0.300271 105 143.900 Trump ( 0.438095 0.561905 ) \*  
## 15) Construction > -1.09675 187 177.300 Trump ( 0.181818 0.818182 )   
## 30) Production < -0.352069 114 135.300 Trump ( 0.280702 0.719298 ) \*  
## 31) Production > -0.352069 73 18.330 Trump ( 0.027397 0.972603 ) \*

predictionstree = predict(treeMod16 , newdata = test16, type = "class")  
mean(predictionstree != test16$TrumpOrClinton)

## [1] 0.0771757

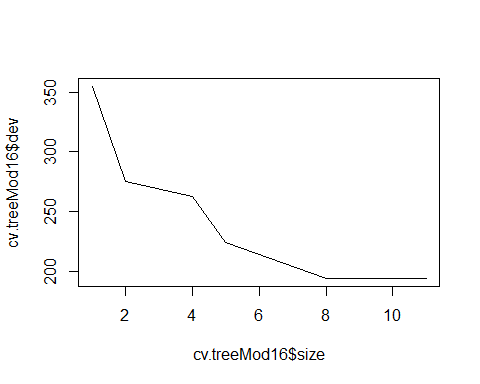
set.seed(1234)  
cv.treeMod16 <- cv.tree(treeMod16, FUN=prune.misclass, K = 10)  
cv.treeMod16

## $size  
## [1] 11 8 5 4 2 1  
##   
## $dev  
## [1] 194 194 224 263 275 355  
##   
## $k  
## [1] -Inf 0.00000 12.66667 20.00000 22.00000 78.00000  
##   
## $method  
## [1] "misclass"  
##   
## attr(,"class")  
## [1] "prune" "tree.sequence"

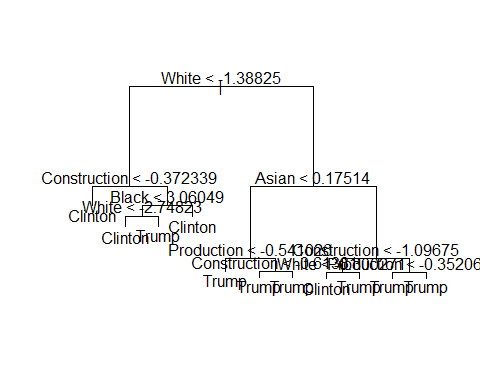
#Pruned tree  
prune.misclass(treeMod16)

## $size  
## [1] 11 8 5 4 2 1  
##   
## $dev  
## [1] 175 175 213 233 277 355  
##   
## $k  
## [1] -Inf 0.00000 12.66667 20.00000 22.00000 78.00000  
##   
## $method  
## [1] "misclass"  
##   
## attr(,"class")  
## [1] "prune" "tree.sequence"

plot(y = cv.treeMod16$dev, x = cv.treeMod16$size, type = "l")



pruneTree <- prune.misclass(treeMod16, best = 9)  
plot(pruneTree)  
text(pruneTree)



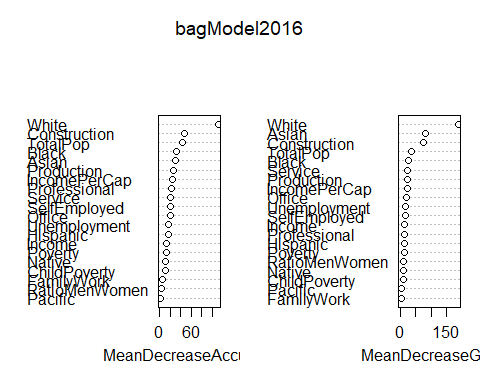
predictions = predict(pruneTree, newdata = test16, type = "class")  
mean(predictions != test16$TrumpOrClinton)

## [1] 0.0771757

#Bagging  
  
set.seed(1234)  
bagModel2016 <- randomForest(TrumpOrClinton ~ ., data = training16, mtry = 20,   
 importance = T)  
bagModel2016$importance

## Clinton Trump MeanDecreaseAccuracy MeanDecreaseGini  
## TotalPop 0.035142900 3.753959e-02 0.0371882142 36.114255  
## Hispanic 0.004947759 5.126874e-03 0.0051110716 13.301849  
## White 0.207709952 4.709684e-02 0.0703774020 188.028414  
## Black 0.020490736 1.426671e-02 0.0151745306 25.285443  
## Native 0.002663399 2.083515e-03 0.0021654249 9.739582  
## Asian 0.073100060 7.206450e-03 0.0166964389 80.791245  
## Pacific 0.002067385 -1.765814e-04 0.0001474786 3.361223  
## Income 0.007036812 4.678152e-03 0.0050302599 14.246129  
## IncomePerCap 0.030590895 8.101118e-03 0.0113227936 21.122116  
## Poverty -0.001919074 4.819396e-03 0.0038363364 12.277723  
## ChildPoverty 0.001005216 2.245919e-03 0.0020709488 8.947170  
## Professional 0.012935669 1.585564e-02 0.0154215258 13.363562  
## Service 0.006840527 7.073023e-03 0.0070428262 22.043828  
## Office -0.002391160 6.586934e-03 0.0052789458 19.621942  
## Construction 0.087069034 1.358233e-02 0.0242242195 74.476661  
## Production 0.040158007 1.112437e-02 0.0152987453 21.301352  
## SelfEmployed 0.012605114 4.745369e-03 0.0058834567 15.183456  
## FamilyWork 0.003461711 8.010619e-05 0.0005654860 3.236774  
## Unemployment 0.011979343 2.874125e-03 0.0041980760 15.185477  
## RatioMenWomen 0.001365478 3.869372e-04 0.0005396061 10.108618

varImpPlot(bagModel2016)



predictionsbag = predict(bagModel2016, newdata = test16, type = "class")  
mean(predictionsbag != test16$TrumpOrClinton)

## [1] 0.05090312

#Random Forest  
  
rfModel5 <- randomForest(TrumpOrClinton ~ ., data = training16, mtry = 5, importance = T)  
rfModel10 <- randomForest(TrumpOrClinton ~ ., data = training16, mtry = 10, importance = T)  
rfModel15 <- randomForest(TrumpOrClinton ~ ., data = training16, mtry = 15, importance = T)  
  
  
predictionsrf10 = predict(rfModel10, newdata = test16, type = "class")  
mean(predictionsrf10 != test16$TrumpOrClinton)

## [1] 0.04926108

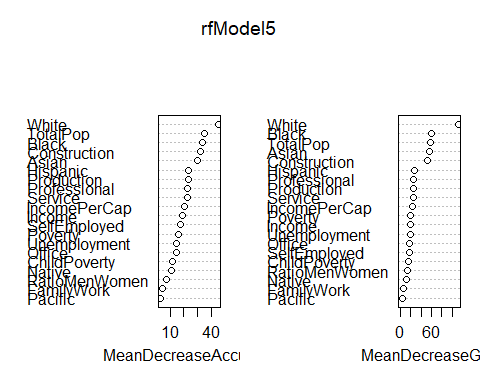
predictionsrf5 = predict(rfModel5, newdata = test16, type = "class")  
mean(predictionsrf5 != test16$TrumpOrClinton)

## [1] 0.05418719

predictionsrf15 = predict(rfModel15, newdata = test16, type = "class")  
mean(predictionsrf15 != test16$TrumpOrClinton)

## [1] 0.04926108

varImpPlot(rfModel5)



#Mtry=10 works best

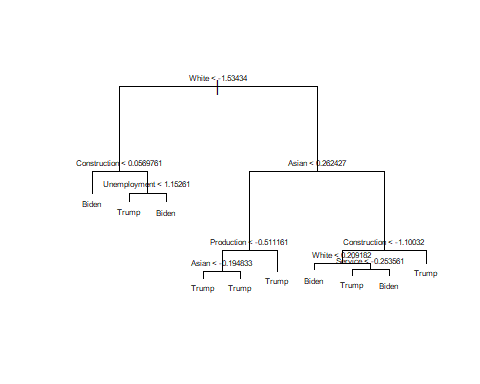
# 2020

set.seed(1234)  
split = sample.split(Trump20Class$TrumpOrBiden, SplitRatio = 0.80)  
training20 = subset(Trump20Class, split == TRUE)  
test20 = subset(Trump20Class, split == FALSE)  
#names(training20)  
# Feature Scaling  
training20[-21] = scale(training20[-21])  
test20 [-21] = scale(test20 [-21])

treeMod20 <- tree(TrumpOrBiden ~ ., data = training20)  
summary(treeMod20)

##   
## Classification tree:  
## tree(formula = TrumpOrBiden ~ ., data = training20)  
## Variables actually used in tree construction:  
## [1] "White" "Construction" "Unemployment" "Asian" "Production"   
## [6] "Service"   
## Number of terminal nodes: 10   
## Residual mean deviance: 0.4183 = 1015 / 2427   
## Misclassification error rate: 0.07058 = 172 / 2437

plot(treeMod20)  
## Can't read the labels very well  
text(treeMod20 , pretty = 0, cex = 0.5)



treeMod20

## node), split, n, deviance, yval, (yprob)  
## \* denotes terminal node  
##   
## 1) root 2437 2096.00 Trump ( 0.15429 0.84571 )   
## 2) White < -1.53434 240 306.90 Biden ( 0.66250 0.33750 )   
## 4) Construction < 0.0569761 135 76.24 Biden ( 0.91852 0.08148 ) \*  
## 5) Construction > 0.0569761 105 133.70 Trump ( 0.33333 0.66667 )   
## 10) Unemployment < 1.15261 73 61.89 Trump ( 0.15068 0.84932 ) \*  
## 11) Unemployment > 1.15261 32 35.99 Biden ( 0.75000 0.25000 ) \*  
## 3) White > -1.53434 2197 1417.00 Trump ( 0.09877 0.90123 )   
## 6) Asian < 0.262427 1896 656.80 Trump ( 0.04167 0.95833 )   
## 12) Production < -0.511161 489 364.10 Trump ( 0.12270 0.87730 )   
## 24) Asian < -0.194833 277 110.90 Trump ( 0.05054 0.94946 ) \*  
## 25) Asian > -0.194833 212 221.80 Trump ( 0.21698 0.78302 ) \*  
## 13) Production > -0.511161 1407 201.30 Trump ( 0.01350 0.98650 ) \*  
## 7) Asian > 0.262427 301 415.20 Trump ( 0.45847 0.54153 )   
## 14) Construction < -1.10032 158 201.60 Biden ( 0.66456 0.33544 )   
## 28) White < 0.209182 111 113.30 Biden ( 0.79279 0.20721 ) \*  
## 29) White > 0.209182 47 61.51 Trump ( 0.36170 0.63830 )   
## 58) Service < -0.253561 31 23.84 Trump ( 0.12903 0.87097 ) \*  
## 59) Service > -0.253561 16 15.44 Biden ( 0.81250 0.18750 ) \*  
## 15) Construction > -1.10032 143 154.50 Trump ( 0.23077 0.76923 ) \*

predictionstree = predict(treeMod20 , newdata = test20 , type = "class")  
mean(predictionstree != test20 $TrumpOrBiden)

## [1] 0.07389163

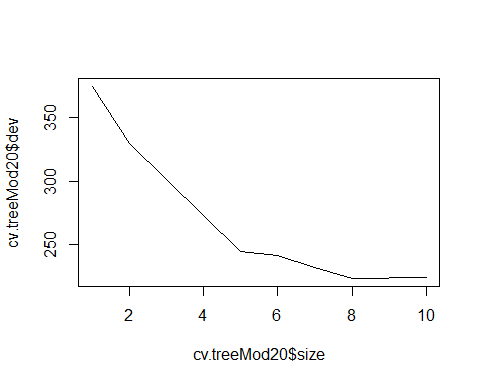
set.seed(1234)  
cv.treeMod20 <- cv.tree(treeMod20, FUN=prune.misclass, K = 10)  
cv.treeMod20

## $size  
## [1] 10 8 7 6 5 3 2 1  
##   
## $dev  
## [1] 224 223 232 241 244 301 330 375  
##   
## $k  
## [1] -Inf 0 10 13 16 26 35 78  
##   
## $method  
## [1] "misclass"  
##   
## attr(,"class")  
## [1] "prune" "tree.sequence"

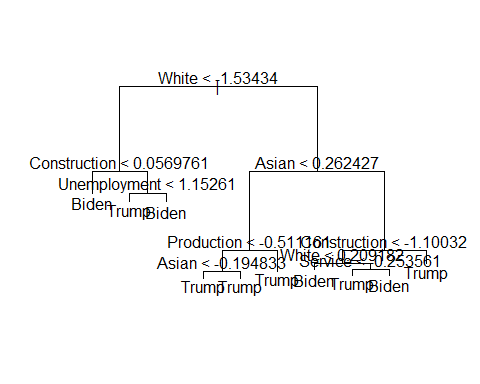
#Pruned tree  
prune.misclass(treeMod20)

## $size  
## [1] 10 8 7 6 5 3 2 1  
##   
## $dev  
## [1] 172 172 182 195 211 263 298 376  
##   
## $k  
## [1] -Inf 0 10 13 16 26 35 78  
##   
## $method  
## [1] "misclass"  
##   
## attr(,"class")  
## [1] "prune" "tree.sequence"

plot(y = cv.treeMod20$dev, x = cv.treeMod20$size, type = "l")



pruneTree <- prune.misclass(treeMod20, best = 9)  
plot(pruneTree)  
text(pruneTree)



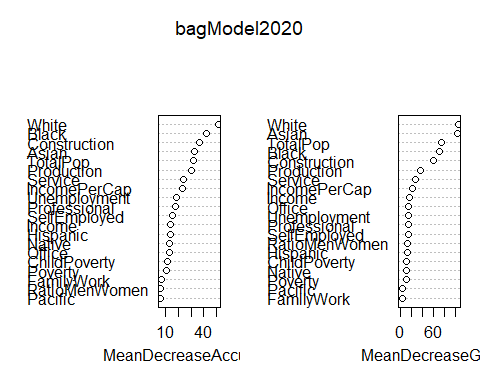
predictions = predict(pruneTree, newdata = test20 , type = "class")  
mean(predictions != test20 $TrumpOrBiden)

## [1] 0.07389163

#Bagging  
  
set.seed(1234)  
bagModel2020 <- randomForest(TrumpOrBiden ~ ., data = training20, mtry = 20, importance = T)  
bagModel2020$importance

## Biden Trump MeanDecreaseAccuracy MeanDecreaseGini  
## TotalPop 0.0616905430 0.0214596583 0.0276827204 74.023336  
## Hispanic 0.0099582420 0.0022417986 0.0034272805 12.302520  
## White 0.1435487445 0.0288062487 0.0465707653 104.683013  
## Black 0.0700842402 0.0216622107 0.0291517610 70.828583  
## Native 0.0065217536 0.0018651563 0.0025848309 10.865380  
## Asian 0.0956301236 0.0100227950 0.0232701687 102.605877  
## Pacific 0.0018884437 0.0002336896 0.0004895237 3.901842  
## Income -0.0002345777 0.0076892980 0.0064534444 15.471783  
## IncomePerCap 0.0151666011 0.0130642422 0.0133963960 21.251979  
## Poverty -0.0025584726 0.0039710048 0.0029575540 10.368057  
## ChildPoverty 0.0083107491 0.0017314506 0.0027448571 11.387814  
## Professional 0.0240966043 0.0084860565 0.0108855794 14.439320  
## Service 0.0120231649 0.0075643651 0.0082483609 26.536276  
## Office -0.0034573960 0.0035120637 0.0024323060 14.913477  
## Construction 0.0745897261 0.0111593115 0.0209963438 59.736861  
## Production 0.0503652881 0.0087910115 0.0152175425 35.394330  
## SelfEmployed 0.0083050984 0.0029097156 0.0037440065 14.342388  
## FamilyWork 0.0055213497 -0.0001948857 0.0006849961 3.656885  
## Unemployment 0.0144614997 0.0026629371 0.0044889546 14.880151  
## RatioMenWomen 0.0043148517 0.0002908379 0.0008984820 13.288054

varImpPlot(bagModel2020)



predictionsbag = predict(bagModel2020, newdata = test20 , type = "class")  
mean(predictionsbag != test20$TrumpOrBiden)

## [1] 0.06732348

#Random Forest  
  
rfModel52020 <- randomForest(TrumpOrBiden ~ ., data = training20, mtry = 5, importance = T)  
rfModel10 <- randomForest(TrumpOrBiden ~ ., data = training20, mtry = 10, importance = T)  
rfModel15 <- randomForest(TrumpOrBiden ~ ., data = training20, mtry = 15, importance = T)  
  
  
predictionsrf10 = predict(rfModel10, newdata = test20 , type = "class")  
mean(predictionsrf10 != test20 $TrumpOrBiden)

## [1] 0.06732348

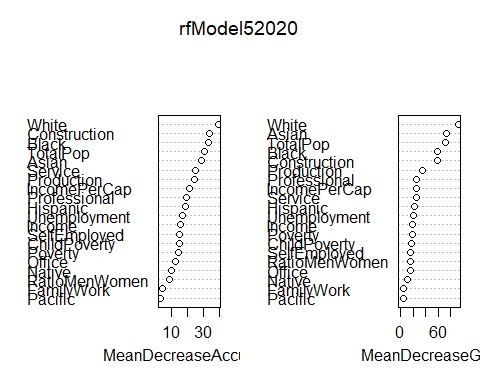
predictionsrf5 = predict(rfModel52020, newdata = test20 , type = "class")  
mean(predictionsrf5 != test20 $TrumpOrBiden)

## [1] 0.06568144

predictionsrf15 = predict(rfModel15, newdata = test20 , type = "class")  
mean(predictionsrf15 != test20$TrumpOrBiden)

## [1] 0.07060755

varImpPlot(rfModel52020)



#Mtry = 5 works best